AD-775 219

REPORT OF THE CACTOS PROJECT: INVESTIGATION OF COMPUTATION AND COMMUNICATION TRADE-OFFS IN MILITARY COMMAND AND CONTROL SYSTEMS

G. M. Cady, et al

System Development Corporation

Prepared for:

Department of the Army

1 September 1972

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE
5285 Port Royal Road, Springfield Va. 22151

DISCLAIMER NOTICE

AD775219

TM-4743/013/00

FINAL REPORT OF THE CACTOS PROJECT: INVESTIGATION OF COMPUTATION AND COMMUNICATION TRADE-OFFS IN MILITARY COMMAND AND CONTROL SYSTEMS

1 SEPTEMBER 1972

G. M. Cady
B. P. Lientz
M. Pillon
N. E. Willmorth

THIS REPORT WAS PRODUCED BY SDC IN PERFORMANCE OF CONTRACT NO. DAHC15-67-C-0149, ARPA ORDER NO. 1327, AMENDMENT NO. 4, PROJECT NO. 2030, PROGRAM CODE NO. 2P10.

77

ABSTRACT

This report describes a group of experiments whose purpose was the expansion, refinement, and validation of the CACTOS computer-network analysis model. The experiments were conducted on both existing and planned computer networks in order to arrive at conclusions with respect to computation resources and to obtain guidelines for use in the design, construction, and modification of computer networks.

The primary issue investigated was the question of whether, in general, centralized or decentralized (distributed) computational power offers the best potential performance and cost-effectiveness for present and future computer network configurations. The conclusion was that partial decentralization, using large computers to achieve economies of scale, provides optimum results for the types of computer networks that will be constructed to meet the needs of the Department of Defense during the 1975-1980 time period. ( )

TABLE OF CONTENTS

Preceding page blank

FIGURES

TABLES

1. INTRODUCTION

This document describes the final results and analyses undertaken in the CACTOS (Computation and Communication Trade-Off Studies) Project supported by the Advanced Research Projects Agency (ARPA). The goal of CACTOS is to analyze, and determine the relationships among, the parameters of computer networks and the performance measures of such networks. The parameters include the hardware and software computation parameters as well as those associated with the communication network intertying the computer sites. The results of this study are intended to be employed by Department of Defense (DoD) agencies in the planning, design, improvement, and modification of military computer networks.

These objectives have been achieved and are described herein. One aim of the Project was to analyze existing networks. The other to determine the parameters and performance measures of importance, as well as the future requirements for information handling in DoD agencies, with regard to planned networks. To perform the trade-off analyses in a quantitative manner, an analytic modeling tool was constructed. Programmed in FORTRAN IV, the model is capable of handling both computation and communication parameters. It is described in detail in Appendix A, along with the underlying equations, information flow, and assumptions.

In order to be of utility, a network analysis tool must be validated for both computation and communication analyses. The validation of the software and hardware computation analysis is described in Appendix B; the communication analysis was drawn from previous work, and its validation was discussed in an interim report. Validation was also performed using several existing systems considered early in the study.

The major analysis tasks of the CACTOS Project are described in Sections 2 and 3. The two sets of experiments for obtaining the general relationships are described in Section 2.1. In Section 2.2, the results of the first set of

experiments are reviewed. In particular, trade-offs were performed among distributed and centralized processing, effect of core and job size, and job type as measured by the average percentage of time jobs were in CPU versus I/O operations. It was found that a configuration of large computers distributed in a semi-centralized configuration was more cost-beneficial than either widely distributed processing with more, smaller machines or a centralized large computer concentration; this preference held for almost all realistic parameter values. Cost-effectiveness was viewed as the ratio of workload to the product of monthly total cost and average total response time. The superior performance of a CPU-oriented (e.g., scientific) network was also demonstrated. The most effective core sizes tended to be small or medium.

The second set of experiments (the results of which are described in Section 2.3) was oriented toward obtaining guidelines in the construction, design, and modification of networks. In these experiments, communication lines were removed from a completely connected configuration in a stepwise fashion based on the criteria of least loaded or least cost-effective lines. It was shown that these procedures can lead to more cost-beneficial configurations than some typical configurations, such as rings and stars.

The analysis in Section 3 confirms some of the analytic conclusions of Section 2. This section presents the CACTOS systems analysis work for one present and two projected networks. These analyses have a broad scope. The first network considered was the Marine Corps Personnel System (JUMPS/MMS), which is connected through AUTODIN. The CACTOS analysis revealed how system performance could be enhanced by redistributing some of the data bases and logic of the network. The second network analyzed was the Air Force's proposed Advanced Logistics System. Here, optimal channel capacities were computed along with measurements of computation. At a level higher than analyses of existing network plans is requirements analyses for new networks whose plans have not yet been developed. To explore these requirements analyses, a General Services Administration request for proposals for a computer network was employed. The analysis

revealed that less than 10% of the fiscal resources should be spent on communication. Furthermore, even with a high percentage of job time in input/output operations, it is most cost-beneficial to spend as much in the central processing unit (as opposed to core) as possible.

The optimal configuration was semi-centralized, with two main computer centers spread across the United States. These results are in close agreement with those in Section 2.2. Both sets of experiments revealed the same fiscal percentages in communication and CPU for the optimal configuration.

Section 4 presents some recommendations and remarks based on the CACTOS Project results. These recommendations include possible new directions, such as network integration. One result of the CACTOS work, namely the analytic model, is now available for use by other DoD agencies for considering specific conditions and networks.

2. EXPERIMENTS

2.1 FRAMEWORK OF EXPERIMENTS

Two sets of experiments were conducted. Experiment Set I was designed to examine computation characteristics. Experiment Set II was aimed at network design for optimal performance. The framework for each of these is described in 2.1.1 and 2.1.2 respectively.

2.1.1 Experiment Set I

The framework for the experiments whose results are described in Section 2.2 was a 40-node network with centers located in the cities listed in Table 2-1 (the numbers in parentheses indicate the number of centers in the given city).

TABLE 2-1

CITIES FOR EXPERIMENT SET I

Index of City	City	Index of City	City
1	Seattle	14	Denver
2	Buffalo	15	Cincinnati
3	Boston (2)	16	San Francisco
4	Portland	17	Kansas City
5	Milwaukee	18	St. Louis
6	Minneapolis	19	Los Angeles (3)
7	Detroit (2)	20	Phoenix
8	New York (6)	21	Atlanta
9	Chicago (3)	22	San Diego
10	Pittsburgh	23	Dallas
11	Philadelphia (2)	24	New Orleans
12	Cleveland	25	Houston
13	Washington, D. C. (3)	26	Miami

These cities were selected in part from geographic distribution and in part from population and density statistics. Each center was given one or more specified computers tied into the network by a standard modem device. The

network was assumed to be of the message-switching type. The assumptions used to perform the experiments were as follows:

1. Remote and local jobs are both considered. A remote job consists of a message transmission, computation, and a return message. A local job is entirely computation. Messages have single sources and destinations.

2. All messages between two nodes follow the path with the minimum number of links between the nodes (fixed minimum path routing). Ties for the minimum length path are resolved by assignment to the least loaded path.

3. Message and job arrival distributions are assumed to be negative exponential.

4. Interarrival times are independent of message lengths and job sizes.

5. Nodes behave independently of each other. This implies infinite-capacity message buffers.

6. Node switching delays are fixed.

7. Nodes have an infinite traffic capacity.

8. Multiprogramming and multiprocessing are not specifically accounted for in the model.

9. Message transmission is assumed to be error-free. Retransmission is not explicitly taken into account.

Some comments on these assumptions are in order. The fixed minimum path routing has been shown to be close to optimal and requires much less computation than is required for calculations of the optimal case (Frank [3]), although arrival times are more closely approximated by gamma distributions. However, in many cases (especially for summed gamma distributions), the effective differences between the gamma and exponential distributions have been demonstrated to be small. Kleinrock [4] has shown that this occurs when all users on a large network are considered simultaneously. The assumption of infinite-capacity message buffers has been shown to be valid when the network is operating at less than 80% capacity. In a network in an unsaturated state with minimum time delay, the limitations on node capacity are minor (Kleinrock [4]).

The model is described in detail in Appendix A. An overview of the model is given in Figure 2-1. The inputs and outputs to the model are discussed in this section.

The capacity of each communication line was set at 50kb. This channel capacity bears a desirable ratio of communication to computation capacity and reflects the optimum ratio of communication to computation costs found in past experiments of other researchers for second-generation computing experiment. To be representative of third-generation capabilities, the configuration of the network was based on the assumption that the articulation of the network was two--that is, at least two links must be broken to break communication between two centers. The monthly communication cost was based on standard available rates for 50kb line size, given by $15.00 for each of the first 250 miles, $10.50 for each of the next 250 miles, and $7.50 for each mile beyond 500 miles. A minimum cost of $250 per month per line was also assumed.

The general network topology appears as in Figure 2-2. Distances were computed by the model using latitude and longitude data (computer centers in the same city were assumed to be three miles apart). The network's topology was

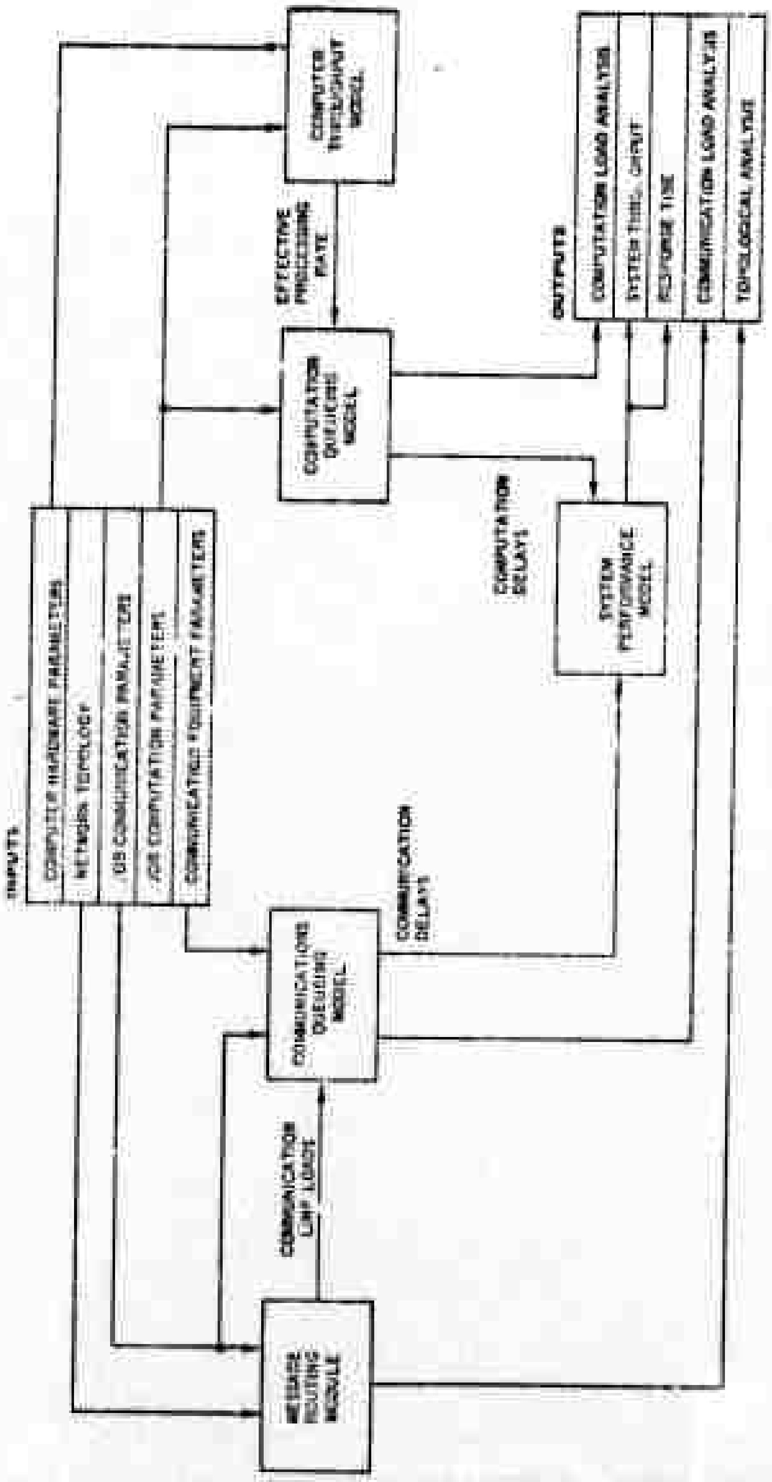


Figure 2-1. The CACTOS Model

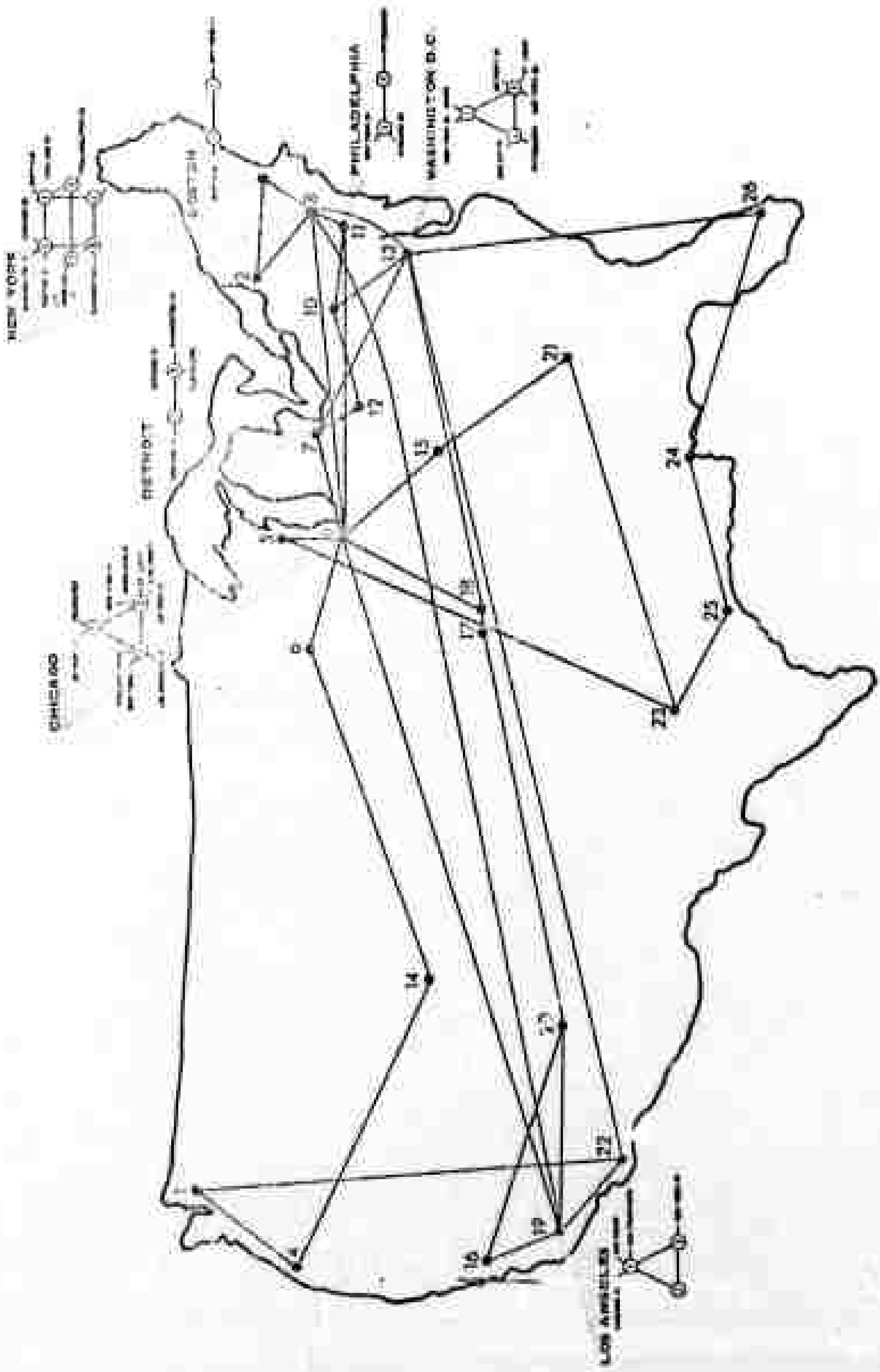


Figure 2-2. Network Configuration--Requirement Set I

constructed iteratively, where at each step the most highly loaded group of links was found. Then a link was added to reduce this load. A lightly loaded link elsewhere in the network was deleted. This proceeded in an enumerative way until no further improvement was possible.

To obtain a representative set of costs and performance, a single hardware manufacturer and generation were selected. This was the IBM 360 line of machines. This line was chosen because of the broad range of compatibly operating equipment and a consistency of costs not yet present in the latest product lines. The machines are listed in Table 2-2.

Since experiments were aimed in part at the type of the job, three basic types of job mixtures were assigned. These, along with their parameters, are given in Table 2-5 in Section 2.1.2. The main parameter here is the percentage of time the average job spends in CPU. This ranges from 90% for scientific to 10% for commercial. To show the sensitivity of computer throughput to core memory size, and to investigate the relationship between job type characteristics and memory, three levels of immediate memory were investigated for each computer. Details on monthly rental price and performance were obtained from Keydata [1] and Auerbach [2].

The cost and configuration information on the computers and peripherals used--360/20, 360/85, and 360/195--appears in Tables 2-2 and 2-3. The 360/85 was assumed to have 2314 disc units, while the 360/195 was given 3330 units. The cost information reflects the costs of peripherals and the mainframe computing unit on a monthly rate.

TABLE 2-2

EXPERIMENT SET I - COMPUTER COSTS

Model	Memory (1000 bytes)	Monthly Cost (000)
360/85	500	$ 85
	1000	150
	2000	200
360/195	1000	215
	2000	258
	4000	300
360/20	16	6

TABLE 2-3

TABLE OF COMPUTER CHARACTERISTICS*

	360/85	360/195
Job processing rate (instruc. microsec.)	6.25	21
Memory size (thousands of bytes)	500, 1000, 2000	1000, 2000, 4000
Word size (bits)	32	32
Disk transfer rate (bytes/microsec.)	312	806
Average disk access time (millisec.)	87.5	38.5
Disk cylinder size (bytes)	146,000	247,600
Average I/O record size (bytes)	7224	7224

The preceding information provides the general topology and cost framework. Remaining to be specified are the job and message characteristics, as well as the specific combinations of hardware for the experiments.

* The 360/20 is included only in communication costs, and its computational characteristics are not included in the experiment.

Three different computation configurations were assumed for the experiments--distributed, semi-centralized, and centralized computing power. The detailed assignments of computers are given in Table 2-4. (The number in parentheses refers to the location within the city). In all cases, the 360/20 computers act as concentrators and message processors. The 360/195 computers are split pairwise in a dual processing mode. In each case, the total raw throughput capacity of the configurations was roughly equivalent (approximately 8 billion modified bits per second). For the 360/196, this takes into account a 15% loss, resulting from executive software overhead in coordinating the dual processors.

TABLE 2-4

EXPERIMENT SET I - COMPUTATION CONFIGURATION

Configurations	Computer Assignment
Distributed	360/85 at all sites
Semi-centralized	360/195 - 2 at each of Boston (2) New York (5) Chicago (3) Washington, D. C. (3) St. Louis Los Angeles (1) 360/20 - other centers
Centralized	360/195 - 2 at each of the 6 New York centers 360/20 - other centers

The mixture of job types is similar to that experienced with a general-purpose computer utility (on-line, interactive operations meshed with remote-job-entry, non-interactive batch processing). These job types were deliberately chosen to emphasize extremes of job mixes and to provide information about the relative merits of centralized and distributed processing power for various job types. Further, the job configurations are directly related to the throughput efficiency of the different computer configurations also being evaluated in these experiments.

The experiments assumed that jobs consist of messages that in turn are decomposed into packets. A packet is the basic unit of bits in the communications part of the experiments. The packet size was set at 2,000 bits.

Several other parameters that had to be determined were job size and the job-arrival matrix. The job arrival matrix has as its (i,j)th entry the number of jobs sent from i to j. The job size was allowed to be variably set at one of three values--5, 10, and 20 megabits. It was recognized that some combinations would be unrealistic. The job-arrival matrix was first set for the distributed case and then centralized as the computer configuration centralized. To derive the number of jobs arising from a given city, the proportion of city population to the total population in all cities was multiplied by the total permissible jobs. The creation of a traffic matrix was based upon the relative distance of the computing centers from the source cities. In the completely centralized case, the job load was distributed equally among the centralized computers. Jobs arising locally around a centralized computer were all assigned to that computer. In the semi-distributed case, where the computers were dispersed to locations about the nation, the traffic was distributed to the processing centers as the square root of the distance from the source city, normalized to the sum of the square root distances to obtain a proportion of traffic. For the completely distributed case, 50% of the jobs arising at a source city were assigned to the computer at that city. The remaining jobs were distributed among all other cities by the square-root distance formula used above. The selection of square root of distance was based on reducing loads between distant cities somewhat but not to an excessive degree. Another assumption of the message traffic was the allowance for acknowledgment messages.

The above framework established a set of 81 distinct experiments in which three values of each of the following parameters were set: job size, configuration, job type, and core size. Total cost of the network configuration varied from 3.1 to 8.3 million dollars per month.

It should be noted that many additional runs were performed to set up the network configuration and its parameters. In particular, experimental runs were required in configuring the network topology itself and in setting parameters to obtain feasible response times. Here response time is infinite and infeasible, since it is impossible to process the jobs in a day.

The absolute level of work was based initially upon an estimated 70% utilization rate of the raw processing power (i.e., total megabits modified per second by all computers) of the system. This total utilization level was adjusted during the experimental runs to reflect the reduced throughput resulting from the assumptions concerning job characteristics.

The assumption that jobs arise in proportion to population has been made in previous network analysis reports in connection with message traffic. Dependence on the populations of cities at both ends of the link can reflect the difference in computing power for major centers. The results of the first set of experiments are examined in Section 2.2.

2.1.2 Experiment Set II

The second set of experiments was focused on the configuration and communications aspects of networking. In the first set of experiments, the network topology was fixed, and computation and message properties varied. Here, the goal was to configure a network based on articulation, reliability, and cost-effectiveness. By following several policies of link deletion from a completely connected configuration, the most cost-effective topology was derived under various constraints of articulation level. The cost-effectiveness measures of the resulting configurations are compared in Section 2.3 with those of ring and star topologies.

The framework of these experiments was more restrictive than that of the first set. Eight nodes were selected in the following cities:

San Francisco	Boston
Los Angeles	New York
Chicago	Philadelphia
Detroit	Washington, D. C.

Initially, a fully connected network topology was assumed, so that any two sites could communicate directly. Line capacity was set, as before, at 50kb. No computation was done at any node, so that the computation processing characteristics were deleted.

The message size was set at 2,000 bits, and the packet size at 1,000 bits. No acknowledgment messages were assumed. Two job-arrival matrices were formed on two bases. The first was the distance-population formula of the first set of experiments. The second was a symmetric traffic matrix where an equal number of jobs was sent between any two nodes. The experimental results are discussed in Section 2.3.

2.2 EXPERIMENTAL RESULTS: SET I

The basic experiments varied the concentration of computing power, job type, core size, and job size. Inference can be made concerning the relationship between system input parameters and network performance measures. The basic network performance measures are cost, response time, job throughput, and measures of cost-effectiveness.

<u>Costs</u> included the entire monthly costs associated with the computer and communication system hardware, including the computer peripherals and memory units, communication interface units (modems, switches, concentrators, etc.), initial costs of the central processing units, and direct channel costs.

Response time was defined as the mean time for both computation and communication processing, often involving the averaging of combinations of times from several computation nodes and communication links. However, initial generation and output distribution times were ignored; only network times were considered.

Throughput was defined as the number of jobs processed per day, modified by considerations of job and message size. Throughput is thus the workload of the system, rather than system capacity.

The principle measure of cost-effectiveness was the quantity throughput per dollar per unit response time. That is, the total throughput (number of jobs times job size) was divided by the product of total monthly cost and the mean total response time.

In interpreting the results, the fact that several network configuration and job characteristic parameters were held constant needs to be kept in mind. Total computation and communication capacity was held roughly constant for all configurations (the capacity for computation was set at 250 million instructions per second, while the line capacities were fixed at 50kb for communication. The network topology, except for the distribution of computing capacity, was fixed (as described in the previous section). Although central memory capacity was varied, all other aspects of the computer facility configuration were held constant for a given computer. Hence, cost varied with the constellation of computing processors and memories used, but not with other (fixed) aspects of the computer or communication configuration.

The results of the 81 experimental runs are given in Table 2-5 in a nested arrangement. The code for concentration, given in column (1), is:

D--Distributed

S--Semi-centralized

C--Centralized

TABLE 2-5

RESULTS OF EXPERIMENTS

(1) CONCEN-TRATION	(2) JOB TYPE	(3) CORE (MB)	(4) 5 MB EXPT. NO.	(5) 5 MB CEx10-2	(6) 10 MB EXPT. NO.	(7) 10 MB C.E.x10-2	(8) 20 MB EXPT. NO.	(9) 20 MB C.E.x10-2
D	S	5	1	19896	28	20034	55	18826
D	S	1	2	13337	29	13743	56	13367
D	S	2	3	11014	30	11506	57	11400
D	M	5	4	08531	31	08730	58	04070
D	M	1	5	05565	32	06242	59	05581
D	M	2	6	04475	33	05259	60	05297
D	C	5	7	03806	34	04719	61	0
D	C	1	8	02292	35	03032	62	03200
D	C	2	9	01763	36	02382	63	02929
S	S	1	10	36235	37	41341	64	43268
S	S	2	11	32794	38	38493	65	41003
S	S	4	12	29631	39	35030	66	37785
S	M	1	13	11870	40	15259	67	15740
S	M	2	14	10421	41	13754	68	15799
S	M	4	15	09212	42	12293	69	14561
S	C	1	16	03769	43	05247	70	05188
S	C	2	17	03241	44	04540	71	04751
S	C	4	18	0359[illegible]	45	03963	72	04191
C	S	1	19	34338	46	35514	73	44019
C	S	2	20	31125	47	36931	74	42027
C	S	4	21	27811	48	33491	75	38843
C	M	1	22	11070	49	14415	76	15553
C	M	2	23	09689	50	12909	77	16447
C	M	4	24	08546	51	11498	78	15301
C	C	1	25	02836	52	03978	79	0
C	C	2	26	02439	53	03429	80	06173
C	C	4	27	02134	54	03003	81	05501

The amount of core memory associated with each main processor is given in megabytes in column (3). Columns (4), (6) and (8) give the experiment number, and columns (5), (7), and (9) the cost-effectiveness (CE) index for each of the 81 experimental runs. These sets correspond to the three job sizes, set one for jobs of 5 megabits (MB), set two for jobs of 10 MB, and set three for jobs of 20 MB. (Note that at the 20-MB job size, two experimental runs--numbers 61 and 79--exceeded the capacity of the system--mean and response time was very long--and a zero cost-effectiveness index is indicated.) Cost-effectiveness is defined here as the square root of the throughput divided by the product of (a) total network costs squared and (b) the mean total response time.

Although some caution must be exercised in interpreting the results since only a few of the myriad possible variables and combinations of variables were manipulated, some conclusions seem clear. For instance, the productivity of a particular configuration is related in a direct and dramatic fashion to the degree of CPU utilization that is achieved. The cost-effectiveness index falls drastically as the fraction of time in computation changes from 90% (scientific) to 50% (mixed) and 10% (commercial). While the inefficiency of I/O-bound jobs is generally accepted in the computation and communication field, that the interrelationship should be so severe was not entirely expected. Although interleaving and time-sharing of jobs may do much to alleviate the inefficiency, it would appear that data-handling techniques may have greatest promise for technological payoff in the future.

A similar instance is the growth of productivity with computing load (job size, in this case). The cost-effectiveness index continues to increase until the system or processor becomes saturated, after which it begins to fall off rapidly. In other contexts, the efficiency of channel utilization in terms of response time has been found to decline, depending upon a variety of circumstances, in the 70%-90% channel-utilization points. While there are not enough data points in this study to make such fine distinctions, the general premise is supported. Since the relationship between load and response

efficiency is well established, the experiments were designed not to reaffirm it but to explore some of the interrelationships between job size and memory size and the resulting cost-effectiveness.

The relationship between job size (load) and memory size seems reasonably clear. The efficiency of core-storage additions increases as the load on the processor increases. That is, from our data, at a relatively light load (jobs of five MB) small core is more cost-effective; at relatively high loads (jobs of 20 MB), larger core becomes more cost-effective. The shift is a gradual one, however; even with loads large enough to swamp computers with relatively small core sizes, a moderate core is more cost-effective than a very large one. This is revealed in Figure 2-3, which graphs job size versus cost-effectiveness for various core sizes. Further investigation of this relationship, of memory mix, of parallelism, and of pipelining seems desirable. However, these would probe somewhat more deeply into processing configurations than is desirable for general networking applications. It might be pointed out that system response time continues to improve with increasing core sizes. It is the disproportionate cost of extra core (in comparison to the increase in cost-effectiveness) that inhibits the strength of the relationship.

Of major interest in this investigation is the relationship between distributed and centralized computing. The difference in relative concentration of computing power between the completely centralized and the semi-distributed cases lies in the distribution of large computers by location around the U. S. That is, the centralized case does not use one super-powerful computer, but a concentration of 12 very large ones, a situation that is duplicated with different locations in the semi-distributed case. The data clearly support the hypothesis of economy of scale; large computers are much more cost-effective than smaller computers (although all computers considered in this study were quite large) and especially so as the load becomes high. That is, with high load into a constant-capacity net, the smaller computers suffered in comparison with the larger. Similar results have been found in communication

MEMORY SIZE COST-EFFECTIVENESS

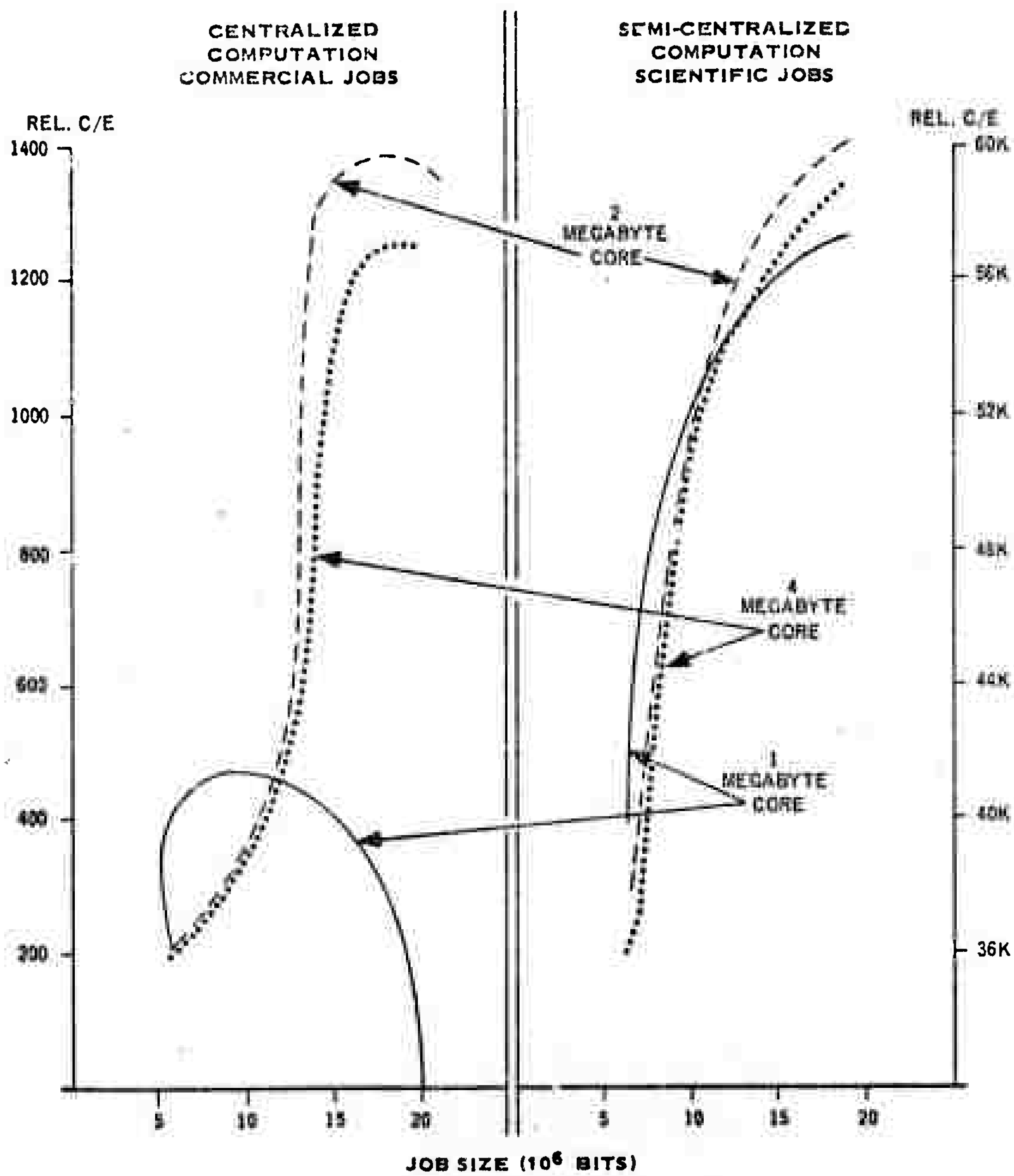


Figure 2-3. Memory Size Cost-Effectiveness

channels, where, for constant capacity, one large line has been found more efficient than a bundle of smaller ones insofar as throughput is concerned. For response time, there is some evidence that multiple channels or processors yield better results in speed but not necessarily in terms of cost-effectiveness.

From the experimental results, the semi-distributed configuration appears more cost-effective than the completely centralized case, partially as a result of shorter, and hence more quickly responding, communication lines. No attempt was made to adjust communication network capacity or cost to accommodate differences in the traffic distribution under the various cases. It is quite possible that further fine-tuning to reduce costs could have been found, or that a better allocation of the capacity might have been found. An anomalous situation does arise in the data for large jobs (20 MB) in that the centralized case seems more cost-effective. This is probably due to the fact that the bulk of the job traffic arises in the Eastern cities, and, by moving computers away from the central moment of traffic sources, costs have been increased. Further investigation of this aspect of network optimization is indicated. Frank and Frisch (1971) and Martin (1972) have indicated approaches to the problem for communication nets. These, in conjunction with resource-allocation algorithms, should provide fairly ready answers to optimal location of processing centers.

There are, of course, several other arguments against complete centralization besides relative cost-effectiveness for throughput. The most relevant of these is the relative vulnerability of a completely centralized facility to the effects of failure of processing or transmission equipment (reliability impacts), environmental effects such as blackouts or brownouts of electrical power, inclement weather, sabotage, or hostile action. On the other hand, larger processors frequently have other advantages, such as faster, more powerful peripheral equipment as well as superior and more powerful central processing units, instruction repertoires, and memories. Larger computers often have

more powerful operating systems and programming languages available to them. Facility operation and maintenance costs of a few central facilities will (or may) also be lower, replacement parts are more easily handled, and record keeping and administration are made easier. Nonetheless, the opportunities for load balancing, the superior failsafe capabilities of alternative locations, and the opportunity for specialization of some computers for specific kinds of jobs with resultant increases in efficiency seem to bolster our general finding that semi-distributed computing provides a superior operation.

2.3 EXPERIMENTAL RESULTS: SET II

The second set of experiments was focused on the near-optimal design of computer networks. The general aim embodied deriving criteria for assisting in automatically generating cost-beneficial network configurations. Another goal was to find the sensitivity between optimization with respect to topology and the communications traffic input data as well as the performance criteria.

Recall from Section 2.1.2 that the configuration was an eight-center network with equal-capacity lines. Costs were based on line costs, and reliabilities were based on time reliabilities. No acknowledgement messages were allowed. The beginning topology for all experiments was a completely connected one in which there was a direct connection between every two centers. Links were removed individually in a sequential manner. Two measures of cost-effectiveness were employed. One is based on throughput factored by cost. The curves in Figure 2-4 are based on this criterion. In this figure, cost is graphed versus bits/second per dollar cost.

Figure 2-5 is based on the cost-effectiveness measure of response time. Here the workload is a constant, so that cost is plotted versus a constant (10^5) divided by the product of mean total response time and monthly cost.

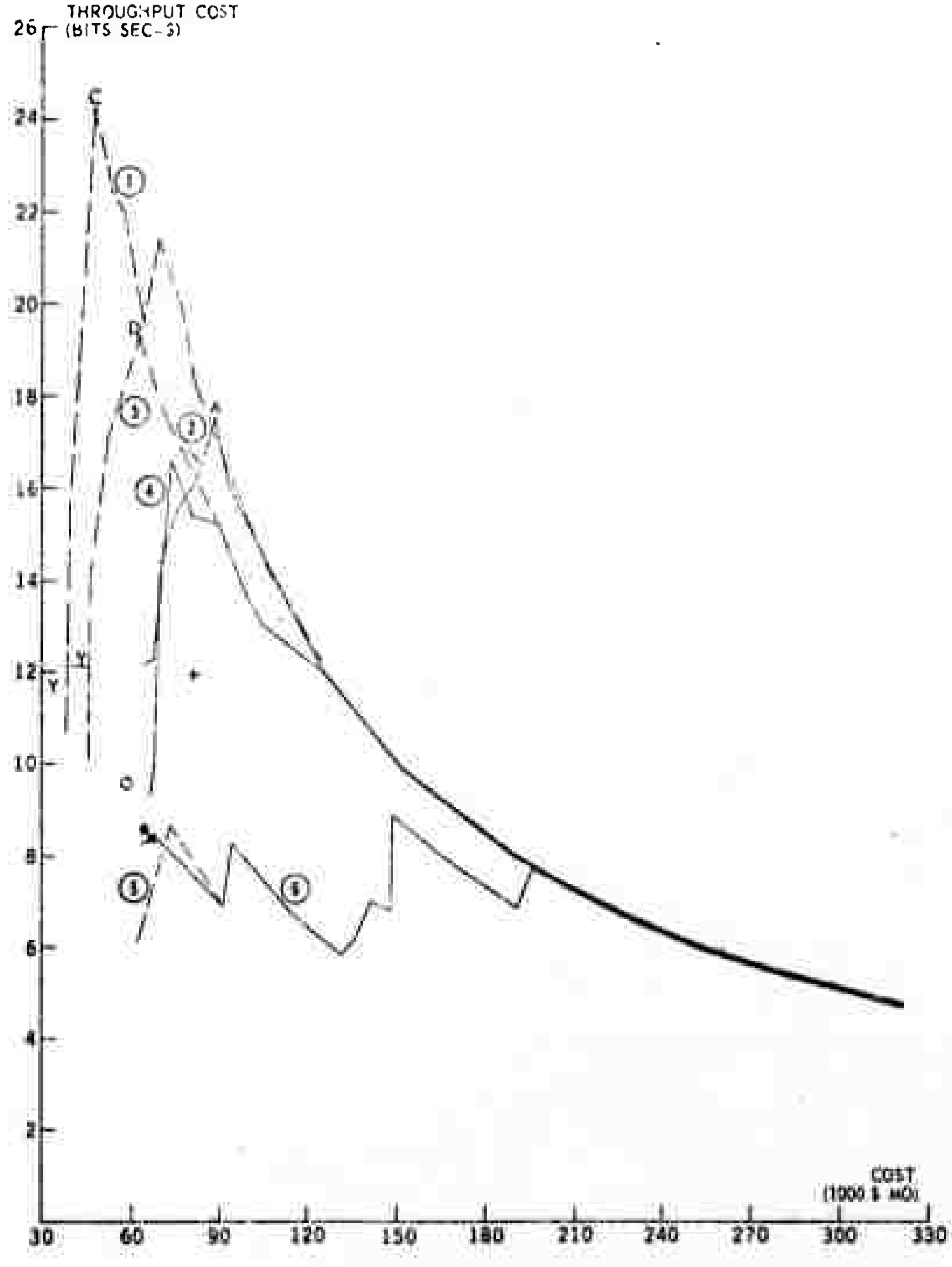


Figure 2-4. Configuration Throughput Cost-Effectiveness

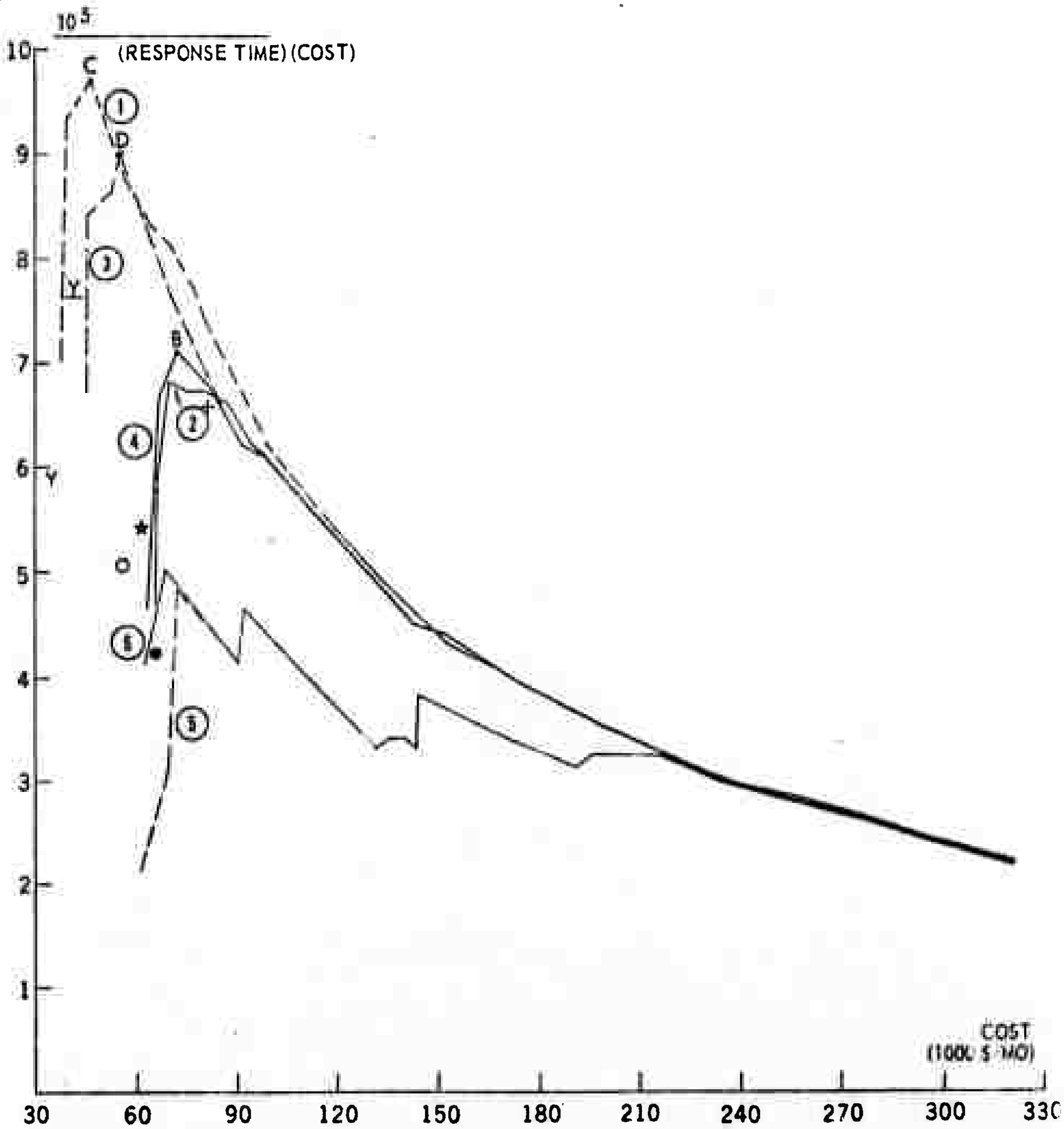
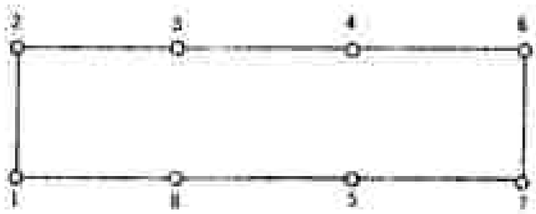
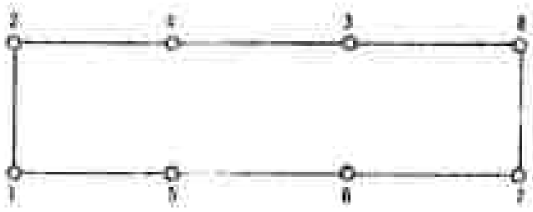


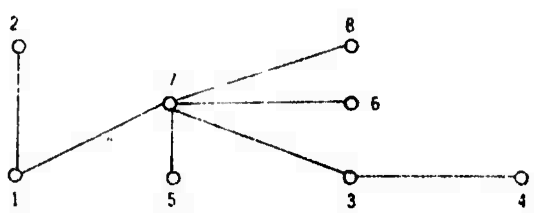
Figure 2-5. Configuration Response Time Cost-Effectiveness



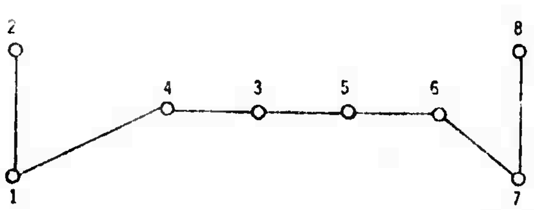
RING CONFIGURATION BASED ON TRAFFIC



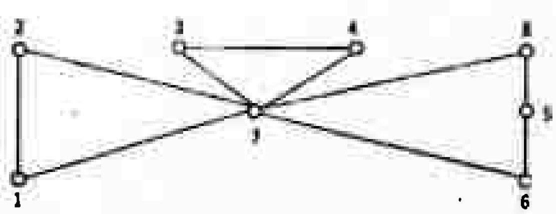
RING CONFIGURATION BASED ON GEOGRAPHY



MINIMAL SPANNING TREE CONFIGURATION BASED ON TRAFFIC

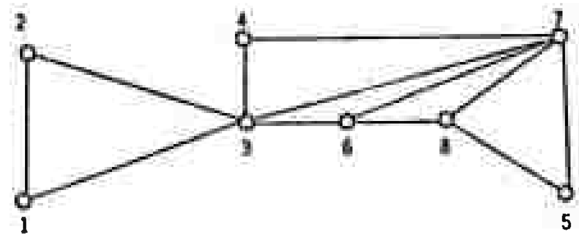


MINIMAL SPANNING TREE CONFIGURATION BASED ON GEOGRAPHY

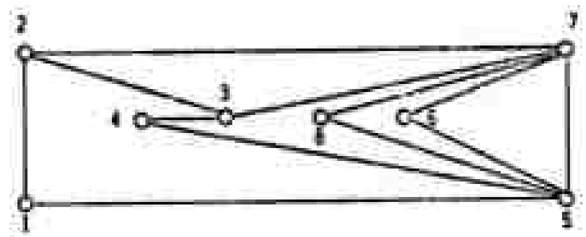


CONFIGURATION BASED ON THE HEAVIEST COMMUNICATING NEIGHBORS

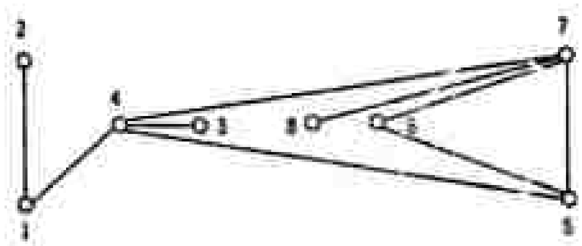
Figure 2-6. Standard Configurations



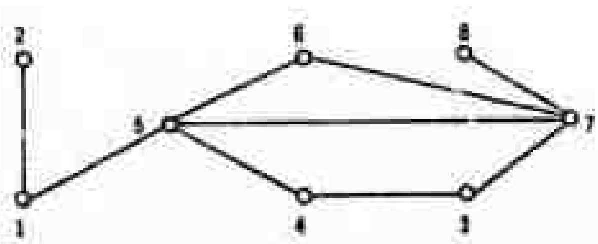
(a) MOST THROUGHPUT COST EFFECTIVE WITH ARTICULATION LEVEL 2 -- POINT A



(b) MOST RESPONSE TIME COST EFFECTIVE WITH ARTICULATION LEVEL 2 -- POINT B



(c) MOST COST EFFECTIVE WITH ARTICULATION LEVEL 1 -- LINK REMOVAL BY COST EFFECTIVENESS -- POINT C



(d) MOST COST EFFECTIVE WITH ARTICULATION LEVEL 1 -- LEAST LOAD LINK REMOVAL -- POINT D

Figure 2-7. Most Cost-Effective Configurations

The topology-determination process of successive link removal through cost-effectiveness may be compared to the results achieved by a priori network specification. A topology can be prespecified, and then the links can be constructed on the basis of either geographic or message-traffic considerations. Two common topologies are a ring and a star. A ring is the configuration with a minimum number of links that gives articulation level two, while a star has a central site and all other sites are connected only to the central site. Another method is to use minimal spanning trees. Distance or link values could be based on geography or traffic. Yet another method is to link each center to the two other centers with which it has the highest message traffic. These standard configurations are displayed in Figure 2-6, and their cost-effectiveness is given in Figures 2-4 and 2-5, using the following symbols:

ring--traffic	●
ring--geography	0
star--center at traffic center	*
min. spanning tree--traffic	Y̲
min. spanning tree--geography	Y
heaviest communicating neighbors	+

In Figures 2-6 and 2-7, the following numerical code for the cities is used.

1	Los Angeles	5	Washington, D.C.
2	San Francisco	6	Philadelphia
3	Detroit	7	New York
4	Chicago	8	Boston

The curves in Figures 2-4 and 2-5 are labeled according to the following code.

Label	Description
1	Link removal by cost-effectiveness, articulation level 1
2	Link removal by cost-effectiveness, articulation level 2
3	Link removal by least loads, articulation level 1
4	Link removal by least loads, articulation level 2
5	Uniform traffic link removal, articulation level 1
6	Uniform traffic link removal, articulation level 2

Link removal by cost-effectiveness means that, at each step, the link that is least cost-effective in terms of throughput per unit cost is removed. Articulation level 2 means that the constraint was applied that the network had to possess link articulation level 2 at each stage of link removal, including the final network.

Another method of link removal was based on removal of the least-loaded link at a given stage.

Uniform traffic link removal refers to the removal of the least loaded link based on uniform traffic statistics. However, cost-effectiveness and the graph values are based on traffic which is dependent upon population and distance.

In Figures 2-4 and 2-5, the points at which some of the curves attain maximum value are of interest. Some of these have been labeled, and the corresponding configurations are given in Figure 2-7.

Two conclusions are evident. First, note that topology determination based on cost-effective and least-loaded considerations yield far better throughput cost-effectiveness (see Figure 2-4) than any standard topology assumed *a priori*.

However, also note that assuming a uniform traffic matrix does no better--in fact, poorer--than standard least-cost topologies. Hence, although superior when good information exists on traffic loads, the methodology is quite sensitive to inaccuracies in traffic estimates. Quite similar results are achieved for response-time effectiveness, except that ranking node interconnections on probable load levels yields fairly good results (see points Y for a minimally articulated net and + for a two-articulated net in Figure 2-5). Thus, if traffic estimates are only good enough to establish probable ranks, that information will still yield better cost-effective performance than ignoring traffic patterns altogether.

The second conclusion is that the optimum configuration depends on the cost-effectiveness measure and the link-removal process. This is supported in the figures by comparing the maximum value with the constraint of articulation level two. In Figure 2-4, this point is labeled A and is attained by link removal based on cost-effectiveness. In Figure 2-5, however, the most cost-effective point is a different point (labeled B) and is obtained by the least-load link-removal process. These two configurations are quite different, as is revealed by comparing Figure 2-7 (a) and 2-7 (b). Since point A in Figure 2-4 (point B in Figure 2-5) corresponds to optimization by throughput (response time in Figure 2-5), throughput and response time are not necessarily optimized in the same configuration.

To consider different link-removal methods with or without the link articulation level being two, it is sufficient to examine either Figure 2-4 or Figure 2-5. The sensitivity of the maximum value attained to the link-removal method is resolved in the distinct curves and points where a maximum value is attained. Thus, it would be necessary to have several methods of link removal on hand for an automated process.

Another conclusion is that all optimization removal procedures depend on the traffic statistics and, in particular, the job-arrival matrix, which gives the

number of jobs being sent between any two centers. This can easily be seen in both Figure 2-4 and 2-5 by examining the uniform-traffic link-removal curves. In both figures, the curves follow a zig-zag path and are dominated by the least-loaded link-removal cases. This is due to the experiments' being structured by one value of a parameter and evaluated on the basis of another value of the same parameter. This emphasizes the importance of accurate message statistics in the network design process. It also points out the importance of meseage statistics on network performance and measurement.

As might be expected, the most cost-effective networks are those that have articulation level one. This is seen in both Figures 2-4 and 2-5, where points C dominate points A and B. Furthermore, the configuration in Figure 2-4 and the traffic statistics based on population and distance reveal that the most cost-effective network configurations are those that are one-connected in remote or light traffic areas.

However, although level-one networks are more desirable from a cost-effectiveness point of view, they are not desirable when reliability is a consideration. A measure of reliability is the expected number of node pairs that will communicate. The failure rate can then be defined as the probability that a pair of nodes will not be able to communicate at a given time. With these definitions, consider the example of Figure 2-8. In the first network, there are 13 links, the articulation level is two, and the failure rate is .00012. However, in the second network the number of links is 12, the articulation level is one, and the failure rate is .00681. In this particular example, by deleting the link from node 2 to node 7, the failure rate increased by a factor of 57. This is just one example of the articulation level. The exact dependence and increase in failure rate would depend on the network structure.

To summarize this section, we note that articulation level one is desirable but not possible from a reliability point of view. Furthermore, a link-removal method such as is described here is better than a standard configuration

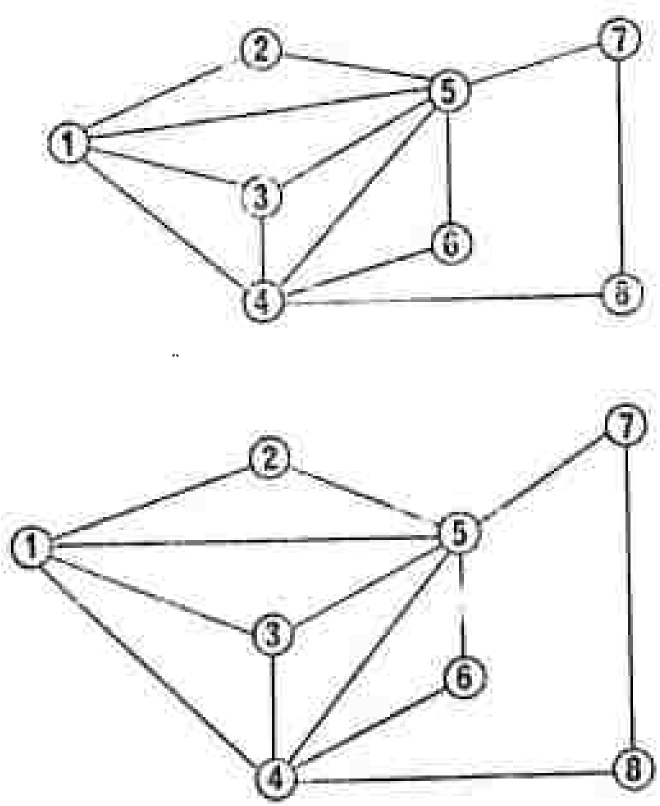


Figure 2-8. Sample Networks--Reliability and Articulation Level

because it is more sensitive to the structure of the network. However, to find the most cost-effective configuration, several methods of link elimination must be tried. When performing the analysis, statistics as close to the estimated or actual traffic as possible must be used for good results.

2.4 REFERENCES

[1] Keydata, Computer Characteristics Review, Waterton, Mass; Keydata Corp., V. 11, No. 1, 1971.

[2] Auerbach Computer Characteristics Digest, Philadelphia; Auerbach Info., Inc., June 1972.

[3] Frank, H., Minimum Link Traffic Routing, unpublished, 1970.

[4] Kleinrock, L., Communication Nets: Stochastic Message Flow and Delay, New York; McGraw-Hill, 1964.

3. SYSTEMS ANALYSIS

3.1 INTRODUCTION

This section examines several systems analysis tasks related to existing and projected computer networks. The purpose of these experiments was several-fold. First, the network analyses could be used to validate the CACTOS model, especially on the communication scale. Second, some trade-off relationships could be discerned. Third, through these experiments, the CACTOS work could become relevant to the near-term planning objectives of DoD agencies. Another benefit of the analyses was to make known some of the analytical methods necessary for a quantitative system view.

The analyses ranged from an existing Marine Corps manpower system (JUMPS/MMS) to a projected system for the Air Force Logistics Command and a modified version of the projected GSA network.

3.2 MARINE CORPS PERSONNEL SYSTEM[1]

The Marine Corps Joint Uniform Military Pay System/Manpower Management System (JUMPS/MMS) is centered in the Marine Corps Automated Service Center (MCASC) in Kansas City, Missouri, with satellite Data Processing Installations (DPIs) at seven Marine Corps bases in the continental United States and overseas. (Initial simulation runs were made using data from eight locations, but the DPI in Danang, Vietnam, has since been phased out.)

The goal of JUMPS/MMS includes the improved management of manpower appropriation and distribution. The details of the system are described by Willmorth [2]. The network is shown in Figure 3-1, along with the basic computer and AUTODIN connections. The main center is at Kansas City at the MCASC. The

[1] This part of the CACTOS project is deeply indebted to United States Marine Corps personnel, especially Colonel J. Marsh and Lt. Col. V. Albers.

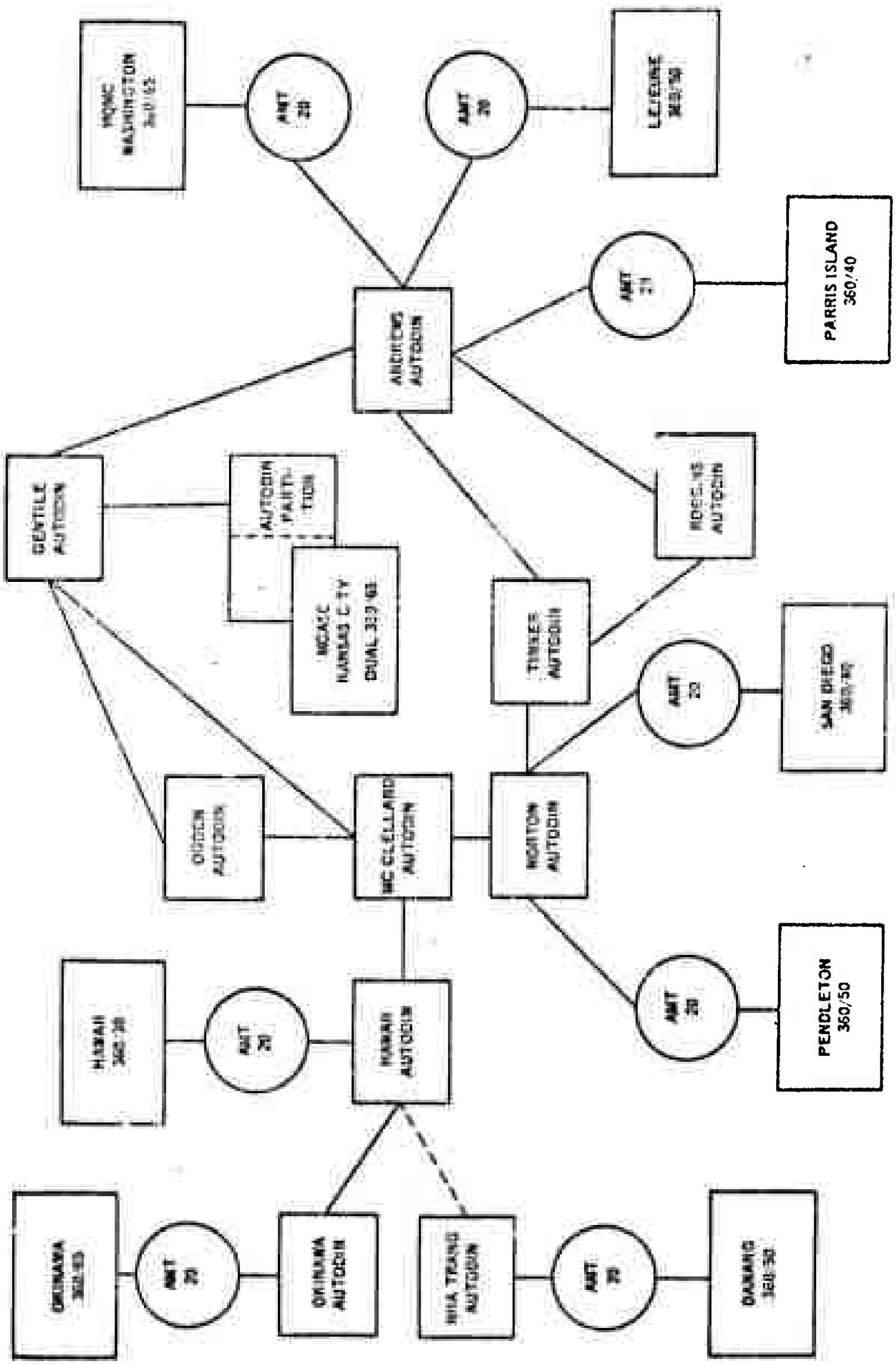


Figure 3-1. The Marine Manpower System

network operates in a batch mode. Problems that come about are due to the priority of personnel message traffic in AUTODIN and at the Marine Corps sites.

The analyses for the JUMPS/MMS system were aimed at several goals. First, the personnel system could be used to validate the CACTOS model. This was accomplished successfully. The model and actual operations statistics differed by less than 5%. The details are described by Willmorth [2].

The second objective was to evaluate the network in terms of improving response time. The response cycle in processing a change through the network was 5 to 10 days on the average. Analyses were performed that showed a 5-10% reduction in time delay by moving part of the data base from Kansas City to a USMC training center. The data on personnel in training prior to duty assignment would then be maintained outside of the home base of the system. In this example, because of queueing delays in AUTODIN and at Kansas City, load sharing of file updating did not significantly improve the system performance, within the constraints of circuit switching and priority levels.

A third set of analyses was performed to determine the effects of increasing the priority level of some or all of the personnel traffic. This is shown in Table 3-1. In this table the response times for the network are given for 10%, 20%, and 30% of the personnel messages having a higher priority. This would be the case in an emergency or exercise deployment of Marine Corps personnel and might occur periodically in restaffing and reassignment.

Analyses were also run to determine the effects of a dedicated personnel network with the present configuration at each USMC site. The results are given in Table 3-2. (The shared column assumes a 90% load factor.) In this table, the single message response time decreases by over half. Total response for the network and MCASC experienced a similar reduction. This indirectly shows the multiple effects of message switching and higher priority levels

TABLE 3-1

PRIORITY DIVISION OF JOBS

(Hours:Minutes)

	MCASC	Net Average
PRESENT LOAD		
Communication	19:25	19:25
Computation	23:34	27:04
Total Response	42:59	46:29
PRIORITY HANDLING		
10% PRIORITY JOBS		
Priority Messages	:20	:20
Priority Jobs	2:46	2:50
Priority Response	3:06	3:10
Remain Messages	19:05	19:05
Remain Jobs	20:22	21:16
Elapsed Time	42:33	43:31
20% PRIORITY JOBS		
Priority Messages	:38	:38
Priority Jobs	4:45	4:02
Priority Response	5:23	4:40
Remain Messages	18:57	18:57
Remain Jobs	17:57	18:06
Elapsed Time	42:17	41:43
30% PRIORITY JOBS		
Priority Messages	1:16	1:16
Priority Jobs	7:22	6:09
Priority Response	8:38	7:25
Remain Messages	17:27	17:27
Remain Jobs	15:40	15:40
Elapsed Time	41:45	40:32

TABLE 3-2

OPERATING TIMES FOR DEDICATED VS. SHARED OPERATIONS

(Hours:Minutes)

	Dedicated MMS Only	Shared 90% Load
Single Message	:11	:37
All Messages	1:10	19:25
MCASC Average		
Single Job	:31	1:33
All Jobs	20:42	23:34
Network Average		
Single Job	1:06	3:59
All Jobs	16:50	27:04
Total Response		
MCASC	21:52	42:59
Network	18:00	46:29

for the traffic. In terms of cost analysis in a dedicated system, communications costs including modems would be between 10% and 15% of the total network cost. The exact figure would depend on the terms and conditions of existing hardware and dedicated communication lines.

To conclude this subsection, we note that the model was validated using actual AUTODIN statistics. Secondly, load sharing within the present environment produced only marginal improvement. Third, a dedicated message switching system or higher priorities in AUTODIN produce increases in performance by substantially reducing response time.

3.3 ADVANCED LOGISTICS SYSTEM

The Marine Corps Personnel System is a centralized network. In contrast, an analysis was performed on a planned decentralized configuration of the AFLC Advanced Logistics System (ALS). Analysis here was aided by the forecasted traffic loads developed by Turhaly and Palmer [1], who conducted transmission simulation on individual lines. Whereas the Marine system was based on current switching, the ALS was focused on message switching.

This study considered the ALS in a general analytic framework wherein the six data centers were considered nodes in the network. The results described below in terms of response time are probably low, owing to the omission of message processing devices. The six bases considered were WPAFB, WRAMA, OCAMA, SCAMA, OOAMA, and SAAMA. The configurations evaluated included those of Turhaly and Palmer and a ring configuration. Distances were obtained from Great Circle distance grids.

Channel capacities were based on anticipated loads reduced to accommodate header messages. Acknowledgement traffic was allowed. The use of fixed-path routing increased some of the capacity assignments above the estimate in the ALS study. The capacities are given in Table 3-3.

TABLE 3-3

CHANNEL CAPACITIES

1	WPAFB	4	SMAMA
2	WRAMA	5	OOAMA
3	OCAMA	6	SAAMA

Route	Capacity (KB)	Route	Capacity (KB)	Route	Capacity (KB)
		ring			
1-3	12.0	3-5	19.2	2-6	12.0
1-2	12.0	4-5	9.6	4-6	9.6
		wheel			
1-2	9.6	3-4	7.2	2-6	7.2
1-3	7.2	3-5	4.8	4-5	4.8
2-3	4.8	3-6	9.6	4-6	7.2
		star			
1-3	19.2	5-3	9.6	4-3	19.2
2-3	9.6	6-3	19.2		
		connected			
1-2	2.4	2-5	1.2	4-5	1.2
1-3	2.4	2-6	4.8	2-4	1.2
1-4	3.6	3-4	4.8	4-6	2.4
1-5	2.4	3-5	3.6	5-6	2.4
1-6	4.8	3-6	7.2	2-3	3.6

TABLE 3-4

RESPONSE TIMES FOR PRIORITY LEVELS (ALS)

Priority	Response	Message Size	Job Size (MB)
1	0-10 min.	56 char. inquiry 396 char. response	25
2	10-30 min.	550 char.	50
3	30 min to 2 hr.	550 char.	50
4	2 - 6 hr.	550 char.	50
5	over 6 hr.	1200 char.	1.25

TABLE 3-5

RESPONSE TIMES (ALS)

	Response Time (Secs.)	Total Response Time	Percent Utilization
Priority 1			
Computer	1.160		20
Ring	.260	1.683	4
Wheel	.365	1.890	4
Star	.234	1.630	4
Connected	.750	2.661	5
Priority 2			
Computer	2.907		13
Ring	.674	4.256	4
Wheel	.907	4.210	5
Star	.492	3.890	4
Connected	1.550	6.010	5
Priority 3			
Computer	8.926		36
Ring	.807	10.541	9
Wheel	1.043	11.012	10
Star	.610	10.146	9
Connected	1.758	12.450	12
Priority 4			
Computer	23.907		60
Ring	.959	25.824	13
Wheel	1.210	26.321	16
Star	.715	25.336	12
Connected	2.256	28.819	19
Priority 5			
Computer	.331		30
Ring	5.693	11.537	56
Wheel	5.821	11.791	54
Star	2.014	4.023	40
Connected	8.752	17.652	77

The priority classes, response times, and message size are given in Table 3-4, along with the respective job sizes in megabits. The latter are based on one-second processing of on-line queries and two-to-ten seconds for a batch job. In the analysis, job and message interarrival rates were set equal. Priority traffic was considered in a declining balance scale.

The response times, using the CACTOS model, are given in Table 3-5. These are the times needed to process one input record. Some other properties considered included the average number of links traversed: 3 for a ring net, 1.6 for a star, 1.2 for a wheel, and, of course, 2 for the completely connected case. In Table 3-5, lowest-priority class is on tape files, so that actual response would be increased over the numbers given for a full tape. Utilization rates are also given in Table 3-5. With highest total capacity, it is not unexpected that the connected net has the highest utilization. The increased utilization of the computers at the fourth-priority level indicates that the system is computation bound. In terms of comparing configurations, the wheel appears to be the most suitable in terms of cost-effectiveness. This was based upon commercial rates and the distances as computed between centers. Obviously, the star is the cheapest when backup systems are ignored.

In summary, then, this analysis evaluated several alternative configurations in terms of performance criteria. Priority levels at which the net is computation bound were determined. Using the model, trade-offs were performed on configuration and topology, and channel capacities were computed.

3.4 REQUIREMENTS ANALYSIS EXPERIMENTS

Using a modification of the Request for Proposal issued by the General Services Administration, the CACTOS project undertook the task of performing a requirements analysis to determine the most cost-beneficial dedicated network configuration given the environment of the system. The goal here was to determine

the best percentage of fiscal resources in communication and to find the best mix between CPU and core-related components on the computation side.

The GSA system was envisioned as a national network with nodes in the following cities [3]:

Boston	Kansas City
New York	Ft. Worth
Washington, D. C. (2)	Denver
Atlanta	Auburn, Wash.
Chicago	San Francisco
St. Louis	Houston
Huntsville	

Core requirements were specified as 300 million bytes. The experiments assumed, for a response-time threshold, that 90% of messages had to have a mean total response time of 10 seconds or less. A fiscal ceiling of $65,000 per month for system operation was assumed. The system is assumed to be data-management oriented, so that the major part of the processing is I/O related.

The purpose of the analyses was, first, to determine the most cost-beneficial combination of communication and computation costs, in terms of throughput per unit cost. The second phase was to then determine the percentage of resources devoted to CPU versus core-storage-related components. The analysis results revealed the relationship between throughput and percentage of fiscal resources in communications shown in Figure 3-2. Throughput for Figures 3-2 and 3-3 is measured in multiples of the job arrival matrix (constant x number of jobs). The optimal percentage is less than 10%, which is consistent with other experiments in dedicated systems.

In Figure 3-3, throughput is graphed versus percentage of resources in CPU. Throughput for Figures 3-2 and 3-3 is measured in multiples of the job-arrival matrix (constant x number of jobs) after the communications expenses of 9.3 have been removed from the fiscal threshold. Several graphs are given for various percentages of job division between I/O and CPU (10%, 25%, 50% CPU). These

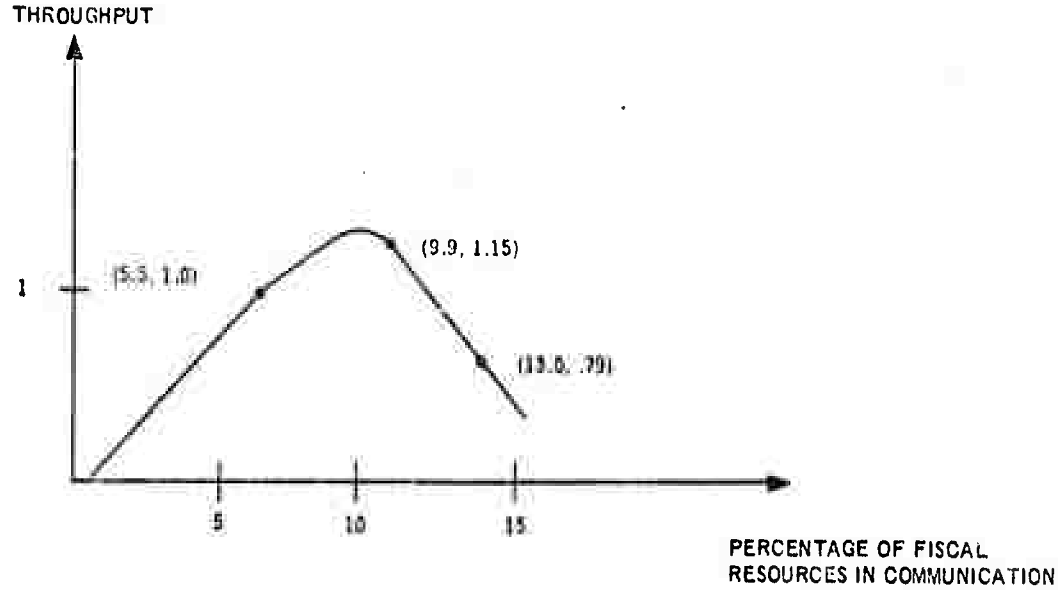


Figure 3.2. Percentage in Communication

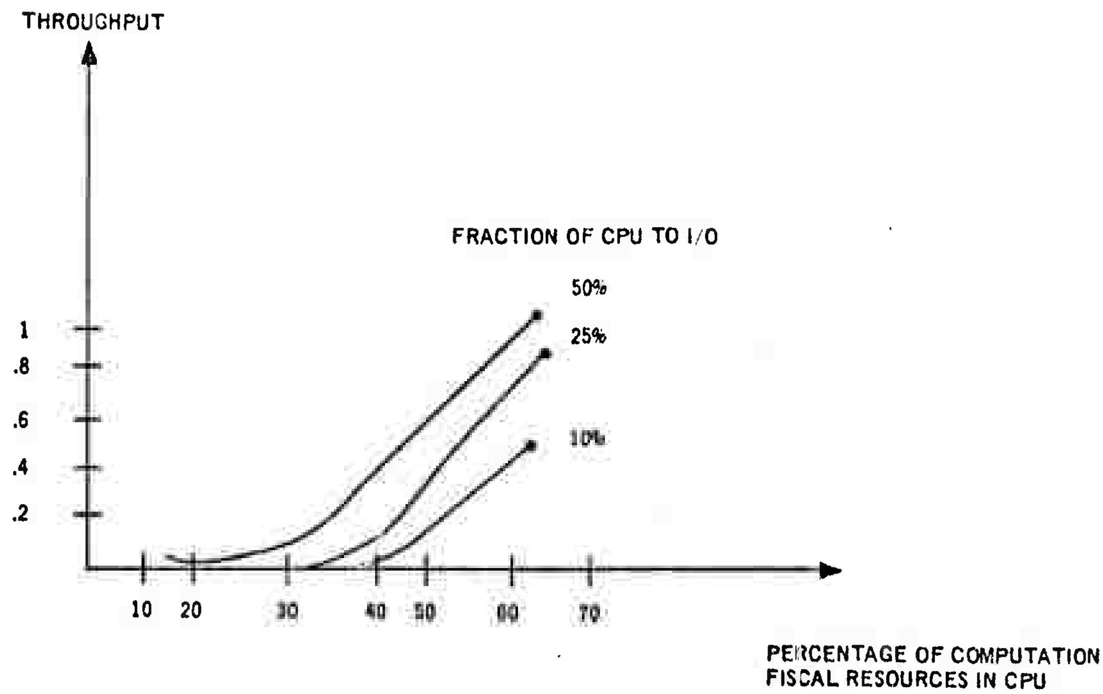


Figure 3.3. Percentage in CPU

reveal that even with an I/O-oriented system, the amount of resources in CPU should be high relative to core-related components. In this case, because only truly attainable configurations are being considered, the maximum feasible percentage in CPU is 62.3%. The reason for this high percentage related, in part, to the fiscal and response time boundaries and also to the dependence of throughput on the CPU-related parameters.

Using IBM third-generation hardware as an example, the optimal configuration consists of 360/65 machines in Denver and at one Washington site, with 360/20 machines at the remaining sites. The configuration can be either a double star clustered at the 360/65 sites or a ring structure. The latter is probably preferable from the standpoint of vulnerability because it has articulation level 2.

3.5 REFERENCES

[1] Turhaly, L. G. and Palmer, J. D. "Advanced Logistics System Communication Network Analysis", Air Force Logistics Command Report, 1971.

[2] Willmorth, N. E. "Report of the CACTOS project: A Preliminary Investigation of Computation and Communication Trade-offs in Military Command and Control Systems", System Development Corporation document TM-4743/012/01, 1 April 1972.

[3] "Request for Proposals for the PBS Information System", General Services Administration, Public Buildings Service, Washington, DC, 1 July 1971.

4. CONCLUSION AND RECOMMENDATIONS

4.1 REMARKS

The basic goal of the CACTOS Project was to develop guidelines for use in the selection and implementation of cost effective computation and communication systems. In striving to achieve this goal, the Project sought to:

- Develop a methodology for describing and analyzing computation and communication systems.
- Determine DoD information processing and transmission needs, as these apply to specific operational needs and functions.
- Investigate the cost-effectiveness of various technological trade-offs.
- Develop optimal planning policies for the design of computation and communication networks and for the incorporation of evolving technology into DoD systems.

In developing a methodology, the Project developed both an analytic and a discrete simulation model of computation and communication networks. In terms of what could be done, the models that have been developed represent a beginning. Original effort has gone into the development of a combined computation-communication model that is available for on-line experimentation. In addition to requiring information on the computing configuration, the model requires careful formulation of other components of an information processing system, which include switches, multiplexors, and man-machine interfaces. While the present model seems adequate for the evaluation of many response-time and queueing questions, it could easily be expanded to include capabilities for examining many other max-flow, min-cost and resource-allocation problems. Considerable effort has been expended in the project on the verification of simulation results against real data. Each expansion of the model should be treated similarly to ensure that simulation results reflect actuality.

In determining DoD information processing needs, the Project discussed data processing problems and systems with many military and governmental agencies and conducted investigations of several existing and planned command and control networks, including the Marine Corps Manpower Management System and the Air Force Advanced Logistics System. Again, in terms of what could be done, this is also just a beginning. DoD is now in the process of developing many new systems and replacing many obsolete systems with up-to-date equipment and procedures. While it is almost certain that all of these will receive a great deal of careful system analysis, it is equally true that most of them could profit from the sort of technological trade-off analysis that a CACTOS project could provide. Unfortunately, neither adequate data processing requirements nor adequate evaluation tools will exist without a considerable research and development effort to provide them.

In investigating the cost-effectiveness of technological trade-offs, the Project investigated, in some depth, the potential trade-offs that were available to system planners. To develop additional technological depth, Project personnel developed a preliminary technological forecast of future developments in computation and communication. Of the many potential trade-offs, the ones that the Project examined deal largely with the economies of scale and the distribution of intelligence (information processing power) within the teleprocessing system. The economies of scale and the economies of technological innovation seem incontrovertible, but the practical implementation of systems that take advantage of these factors is not imminent. A considerable amount of work should be done in the development of practical replacement policies and in the design of new systems. While the Project found evidence that semi-distributed computing networks have some advantages, much more needs to be done in examining the location of processing centers, in allocating functions to various levels in a network, in locating information stores, and in assessing the advantages or disadvantages of specialized processors. In fact, the whole arena of technological trade-offs has hardly been tapped,

and a great deal remains to be done before exact guidance can be given to system designers.

In the development of optimal policies for system design and in the development of efficient replacement policies, a start has been made. The Project reaffirmed some of the trade-offs expressed in the past and formulated some extremely limited laws concerning the interrelationships among computation and communication elements. The development of further system design tools and guides is largely dependent upon the continued evaluation of technological trade-offs. Sharpe [3] has made the initial contribution to the structuring of this field, but a great deal remains to be done. Just in terms of developing cost-performance relationships, the only system components for which reasonable trends seem to be established are central processing units. Even here, a myriad of factors inhibit the declaration of a clear set of principles for predicting system costs and performance. For many other system elements, historical data upon which to base future predictions do not even seem to exist. Developing such trend data is partly inhibited by the ways in which data processing and transmission functions may be combined within a particular piece of system equipment. That is, the development of economic information is dependent in part upon studies of the allocation of functions (e.g., the distribution of intelligence) to various parts of the system, which in turn is influenced by what is known about costs of configuring a system one way or another. A continuation of investigations in this area should be of considerable benefit to the state of the art of teleprocessing systems.

Replacement policy in an era of rapid technological development is certainly a matter of great concern. Roberts [1] has stated some of the considerations that impact a replacement policy for computers, such as the number of years before the acquisition of a new computer, the length of time an old system is to overlap with the new, the growth of the work load, the relative advantages of

lease and purchase, the hidden costs of software, facility and retraining requirements, and the interrelationships among these. Schneidewinde [2] has also formulated a model for predicting optimal replacement of computers, but in both cases these considerations need to be expanded to consider the kinds of multicomputer teleprocessing networks of concern to command and control systems. At present, except in the most simplistic terms, computation and communication system replacement policies cannot be recommended to DoD. It may be obvious to all that many current DoD systems are technologically obsolete and probably economically inefficient, but advising DoD on the policies that it should adopt to keep its systems technologically current and optimally cost-effective is most questionable without further precise formulation and evaluation of technological and procedural trade-offs.

Another trend that needs to be addressed is the increasing degree of integration of information processing networks. There is a proliferation of systems for both the processing and the transmission of information. There is an increasing need for the exchange of data among systems. To the DoD user of information, there is a definite need for the separate information systems to be "transparent" to his use. That is, when he turns to his control and display console, he does not care where the information is stored or what system is processing it. He wants the needed information to be delivered to him without hyperbole in procedure or content. Such system integration, given the plethora of existing systems and the procedures for using them, is more difficult than designing a new system. Ways and means of overcoming system incompatibilities and of establishing data and procedural standards need to be studied.

By and large, DoD is aware of these problems and is approaching them, largely on an individual system basis. It is highly recommended that centralized DDR&E support be given to such study effort, so that DoD-wide policies can be established. Projects such as CACTOS offer a great many benefits to the development of computation and communication systems for command and control applications. Much favorable notice has been given the effort, but much more is

necessary before such a project can truly impact DoD decision making. Some such centralized effort to develop network design guidelines and cost effectiveness evaluation techniques should be established on an ongoing basis to assist DoD system procurement efforts. There is a vast amount of detailed analysis to be done, but these analyses could save DoD a great deal of unnecessary effort and expenditure of funds on suboptimal systems.

4.2 REFERENCES

[1] Roberts, Lawrence G. "Data Processing Technology Forecast", Unpublished, 1969.

[2] Schneidewinde, N.F. "Analytic Model for the Design and Selection of Electronic Digital Systems", University of Southern California, Doctoral Dissertation, 1966.

[3] Sharpe, W.F. The Economics of Computers, New York: Columbia University Press, 1969.

APPENDIX A

COMPUTATION AND COMMUNICATION TRADE-OFF STUDIES: AN ANALYTICAL MODEL OF COMPUTER NETWORKS *

INTRODUCTION

It is probably safe to say that the most vasive and significant development in comer usage during the 1960's was the rise of time-sharing system. The phenomenal accepce of time-sharing as a modus operandi for puter systems is a direct result of the many efits which accrue from the basic technique simultaneously allowing users to share, on a al basis, the total resources of the computer.

It now appears that the development of the 0's which will most closely parallel the timering phenomenon of the 1960's is the rise of puter networks. By computer network is nt a system comprised of two or more comers, usually at different sites, connected ether by communication links, in which comation is a primary function of the system not merely ancillary to the communication ction. (Communication may also be a primary tem function.) Just as time-sharing increased power of the computer through local sharing computer resources, so computer networks can vide another dimension of power to the er's machine" through resource sharing on a e global scale. The resources shared in comer networks include not only hardware facil- es, but data and software as well. The lowing are some of the efficiency gains ievable through the networking of computers:

1. Duplication of hardware facilities can eliminated or greatly reduced. This is ticularly true in networks which include a e variety of computer sizes and types. ess to a remote computer with some feature uired by a user can eliminate the need to chase a similar facility at the user's site.

2. Programs can be made to run on the puters which handle them efficiently, rather n being forced to run on local equipment ch may be poorly designed for a particular blem.

3. Duplication of applications software n site to site can be reduced. This minates the sometimes nasty problem of pro- m transferability among incompatible machines.

4. Electronic and manual transshipment of ge amounts of data, with its associated costs delays, can be eliminated by operating on ote data bases over a network.

5. Queueing and overload problems at certain facilities can be alleviated by load-sharing schemes, whereby jobs are routed to facilities which have lighter loads. This works best, of course, in networks with similar or identical computer facilities at more than one node.

6. Special purpose languages, which--as compiler construction techniques become more sophisticated--appear to be a cost-effective means of solving certain problems, need be implemented on only one computer which is accessible through a network.

7. Overall system reliability can be greatly enhanced if alternate computer facilities can be accessed via a network in the event of a system failure at one node. The topology of the network can be designed so as to minimize the likelihood of system failure due to communication component difficulties, as well.

8. In military and other applications where vulnerability to attack or sabotage is a significant consideration, computer networks with suitable topology characteristics can provide a degree of invulnerability which cannot be achieved by single-site systems.

9. Overall system degradation due to errors or component failure can be "graceful" in a network, where as it might be catastrophic if networking were not part of the system design.

In summary, the user who is communicating with a network of computers can have at his disposal a much more powerful, versatile, efficient, and reliable tool than the user who is restricted to a single computer. For these reasons, and because technological progress has brought the necessary concepts to fruition, a rapid proliferation of computer networks is anticipated in the current decade.

Careful analysis and design of computer networks, therefore, has now become a matter of consummate importance if their full power and cost effectiveness are to be realized. With these considerations in mind, the Department of Defense, through its Advanced Research Projects Agency, has sponsored a broad program of research into the relevant issues. The results of one part of this effort, the Computation and

---

iis was presented at the WESCON Conference in Los Angeles on September 19, 1972.

Communication Trade-off Study (CACTOS), are reported in this paper, with an emphasis on the quantitative analytical tools developed for the study. The development of adequate tools for quantitative analysis of the behavior of computer networks and of the complex interrelationships among the many parameters involved in their design and implementation constitutes an important first step in making the right decisions about computer networks over the next several years---decisions which will have major impact on military, government, corporate and public interests. Such tools are necessary to identify possible mismatches between projected needs and capabilities and to ensure that the proper trade-offs are being made to best serve the needs of the entire computer-using community.

THE CACTOS MODEL

Meaningful analysis of computer networks demands quantitative analytical tools; to this end, the CACTOS analytical model was developed and implemented under System Development Corporation's ADEPT and ICOS time-sharing systems. To allow the user to quickly perform experiments and explore conclusions tentatively inferred from previous calculations, a fast, interactive tool was desired which would allow great flexibility and yet minimize user inputs when the current test case is similar to a previous one; the implementation of the CACTOS model achieves these objectives to a high degree.

The primary performance characteristics of a computer network are its response time (time between transmission of an input from the user's terminal and receipt at the terminal of an output response from the system) and throughput (maximum rate at which the system can perform work). Measures of these characteristics are the principal outputs of the CACTOS model. Although the two parameters are correlated, they are not deterministically related. For example, designing a system to minimize response time for a given cost does not guarantee that throughput will be maximized for that same cost.

Inputs to the model are the values of parameters which describe the communication hardware, computation hardware, and workload, including some software characteristics, of the system under study. Thus, the model does not design systems; the user designs systems and the model helps him by estimating the performance levels of the various alternatives.

Figure 1 is a schematic diagram which depicts the organization of the analytical model itself. At its heart lies the "Communications Queueing Model." This module considers the communications network, its hardware characteristics, its topology, certain characteristics of the communication methodology, and the communication workload. A queueing analysis is performed which computes the average communication delay of the whole system. The message load on each communication link is computed by the message-routing module. A topological analysis is also performed; its primary value is in vulnerability studies because it indicates the minimum number of links and nodes which must be removed from the system to break communication. The topological analysis is also important when one is trying to correlate such topological parameters as radius (distance, in links, from the most central node to a peripheral node), diameter (longest distance between any pair of nodes), and connectivity (minimum number of links connected to a node) of a network with the output performance parameters.

An analogous computation queueing model evaluates the computational load at each node and the overall average delay due to the computational processing and associated queueing. This evaluation considers the effective processing rate of the computer at each node and the frequency and size of jobs to be processed there. The effective processing rate is generated, in turn, by the computer throughput model, which considers both the characteristics of the computing equipment at the node and the software characteristics of the jobs to be processed there.

Finally, the output of the communication and computation queueing models are combined to give the overall response time and throughput values for the entire system.

Assumptions

Before describing the model in any detail, we must dwell, at least briefly, on the assumptions which have been made in its derivation. As is the case with any analytical model, the user must be careful when using it, to be certain that assumptions made in the derivation of the model either are true in his situation or have little effect on the results.

1. There are two types of jobs being processed by the system being modeled: remote jobs and local jobs. A remote job consists of a message (data transmission across one or more links of the network), followed by a computation at the node to which the message was addressed, followed by a return message. A local job consists of a computation only, with no demands on the network's communication resources.

2. Each message has a single source node and a single destination node.

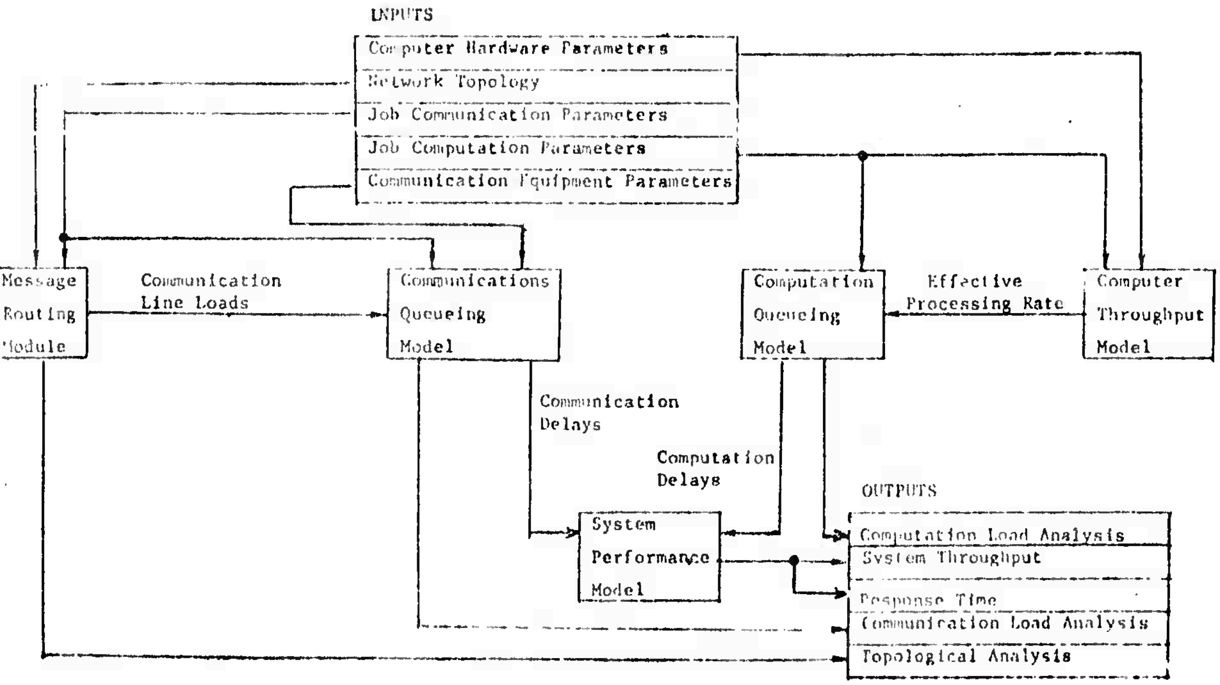


Figure A-1. Information Flow in the CACTOS Model

3. Fixed-minimum-path routing is used. This means that all messages originating at a particular node and destined for another particular node will follow the same path, and this path will be a minimum path (fewest links) between the nodes. In the event that there is more than one minimum path between a pair of nodes, the messages are assigned to the least loaded minimum path at the time of assignment. Experiments have shown that this method of message routing is only slightly inferior to the mathematically optimal method, and that, in fact, the selected routes are generally the same in both methods.[1] The computation of minimum path routes, however, is much easier than that of optimum routes.

4. Message and job arrival rates and sizes are described by negative exponential distributions. Empirical measurements on arrival statistics have tended to substantiate gamma rather than exponential distributions,[2] but the differences have been shown to have small effect on the calculations, and the exponential distribution provides a reasonably good model of typical user requests.

5. Interarrival times are independent of message lengths and job sizes. It is evident that this is a poor assumption if we are describing a single user or processing node, but Kleinrock has gone to great lengths to demonstrate that it is a reasonable description when all users on a sizeable network are considered simultaneously.[3]

6. The various nodes behave independently of one another. This implies, among other things, that there are effectively no limitations on the size of message buffers, for, if a message buffer were to overflow at any node, further transmission of messages to (and through) that node would be blocked, thus destroying the assumption of independent node behavior which our model demands. We have found that the assumption of infinite capacity message buffers is quite valid if the network is operating at 80% or less of its communication capacity. All networks which the CACTOS study has investigated possess this characteristic.

7. In the communications network, the effects of limited node traffic throughput capacity are negligible compared to the corresponding link limitations. In effect, we are assuming that the nodes have an infinite traffic capacity. Past experience has shown that in well-designed networks which are not near saturation and in which time delays have been minimized, node limitations play a minor role.[3]

8. Node switching delays are constant. The switching delay, is, of course, independent of the node's message traffic throughput rate discussed above.

9. All message transmission and computational processes are error-free, so that

retransmissions and recomputations do not occur.

10. All communication is via store-and-forward technology; there is no circuit-switching and no dedicated lines which are unavailable to one or more of the nodes of the network.

11. There is no more than one computer at each node, and no more than one communication link between a pair of nodes. Relaxation of this assumption is planned for the near future.

12. Neither multi-processing nor multi-programming is explicitly accounted for in the model.

The Routing Algorithm

It is important for us to know the volume of message traffic over each link in a network. The frequency of messages on any given link depends on the communication traffic pattern, the network topology, and the routing strategy.

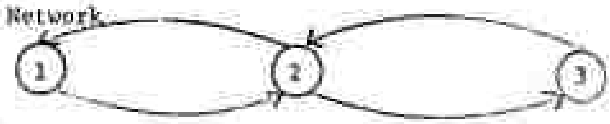
The fixed-minimum-path routing strategy has already been discussed as a model assumption. Network topology and traffic frequency will be represented by matrices. Entries in the connectivity matrix will be defined as $C_{ij} = 1$ if there is a communication link from node i to node j; $C_{ij} = 0$ otherwise. γ_{ij}, the i-jth entry in the job arrival matrix, represents the number of jobs originating at node i to be processed at node j, per time period. (This means that a message will be sent from node i to node j, and a return message will be sent from j to i. Finally, the traffic matrix will be composed of λ_{ij} = frequency of messages across the link from i to j. It is the job of the routing algorithm to build the traffic matrix from the other inputs. Note that if we define the operational (as opposed to topological) average path length to be the average number of links traversed by a message, this quantity is calculable from

$$\text{average path length} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_{ij}}{2 \sum_{i=1}^{N} \sum_{\substack{j=1 \\ i \neq j}}^{N} \gamma_{ij}}$$

where N = number of nodes.

This figure is returned as an output of the CACTOS model and has been found to be a significant system design parameter. Topological parameters of interest which are also "windfalls" from the message routing scheme are the radius and the diameter of the network. A simple example illustrating these ideas is shown in Figure 2.

Network



Inputs

Job arrival matrix: $\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$

Connectivity Matrix: $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

Outputs

Traffic matrix: $\begin{bmatrix} 0 & 4 & 0 \\ 4 & 0 & 4 \\ 0 & 4 & 0 \end{bmatrix}$

Average path length: $\frac{16}{2*6} = \frac{4}{3}$ links

Radius: 1 link

Diameter: 2 links

Figure A-2. An Example of Message Routing Inputs and Outputs

The algorithm selected for traffic routing is a modification of Dijkstra's tree-building algorithm for finding the least-cost paths from one node to all other nodes in a network.[4] In this application, the link cost is artificially set to the number of messages already assigned to a link, plus G, where $G > 2\gamma$. This mode of setting the cost forces the cost-minimizing algorithm to select the shortest path first, and the least-loaded paths second if there is more than one shortest path, which is exactly the scheme desired. The mathematical optimality of Dijkstra's algorithm and the fact that $G > 2\gamma$ guarantee that minimum-link paths will always be selected; however, the second-order balancing may be sensitive to the order in which node pairs are assigned routes. It has been found empirically that imbalances tend to be minimized if all node pairs separated by paths of length one are assigned routes first, all node pairs separated by paths of length two are assigned second, and so on up to the diameter of the network. This scheme is implemented in the CACTOS model.

The Basic Communications Model

The details of a basic model describing the

behavior of a store-and-forward communications network have been described quite clearly and will not be re-derived here.[3,5] Kleinrock's formula for average message delay is:

$$T_{COMM} = \sum_{i=1}^{M} \frac{\lambda_i}{\gamma}\left(\frac{1}{\mu' C_i} + \frac{\lambda_i/\mu C_i}{\mu C_i - \lambda_i} + \frac{L_i}{v} + K\right) + K \qquad (1)$$

where

λ_i = message frequency over link i. (Note that summation here is over links, rather than node pairs, as was done in the previous section.)

γ = overall system message input rate.

$1/\mu'$ = content message's average length.

C_i = channel capacity of link i.

$1/\mu$ = average length of messages, including acknowledgements.

v = propagation rate in the communication links (usually at, or near, the speed of light).

L_i = length of link i.

M = number of links in the network. (Duplex lines are treated as two independent links.)

K = nodal switching delay.

In this expression, $1/\mu' C_i$ is the transmission time, $\frac{\lambda_i/\mu C_i}{\mu C_i - \lambda_i}$ is the queueing delay, and L_i/v is the delay for propagation through the medium of the communication links. The sum of these terms, plus a nodal switching delay K, is weighted by λ_i/γ, which has the effect of multiplying the average delay per link by the operational average path length, or, equivalently, weighting each link's average delay by the amount of traffic which it carries, and then taking an overall average delay for the system. Finally, another K is added in to account for the final switching delay at the destination node.

One word about the difference between μ and μ'. As a technique for error control, many networks require some kind of acknowledgement message to verify each correct transmission. The transmission time for a real message depends only on its own size and the channel capacity; hence μ' is used in calculating transmission delay. Queueing delay, however, depends on the overall loading of a link, including acknowledgement traffic. Since the size of an acknowledgement message is, in general, different from the size of a "content" message, a different average message size, namely $1/\mu$, must be used in the calculation of queueing delay. The operational implementation of the CACTOS model allows the user to choose whether or not the effects of acknowledgement messages are to be taken into consideration.

The interpretation of results from any analytical model must be made in such a fashion as to accurately reflect characteristics of interest in the system being modeled. In the actual use of the message delay model of equation (1), several applications-oriented questions arose. These resulted in some modification of Kleinrock's work to better suit the purpose of the CACTOS study.

Message Size Variability

Messages on different lines of a real network will probably be of different average sizes, and, in fact, the message sizes arising from different sources may fit different statistical distributions.

Kleinrock's equations use standard message size $1/\mu$ and $1/\mu'$ throughout the network; the differences in mean message sizes on different links may be accounted for by merely subscripting μ and μ'. Message sizes are then computed separately for each link in the network and are used separately in the individual calculations of delays on each link. In practice, the traffic going over each link is a function of the original source-destination traffic and message-size matrices and the routing procedure. If individual average message sizes are to be calculated for each link, it is most convenient to save complete information on message traffic assignments as they are fixed by the routing procedures. Thus, if the total number of messages and the total number of message bits are kept for each link in the network as they are assigned by the routing procedure, the mean message sizes, $1/\mu_i$, may be readily calculated.

The degree of sensitivity of the model to this change has not been assessed for any real networks. It will, in all likelihood, be greater in networks with very diverse message loads over the different links. Consider, for example, the network shown in Figure 3. One can imagine a regionalized computation system with this kind of topology, where regional data input centers send short data messages to computational centers. These centers, in turn, accumulate data and then send very large messages to other computation centers for storage or computation. In such a configuration the difference in message sizes over remote

and central links could be a crucial consideration.

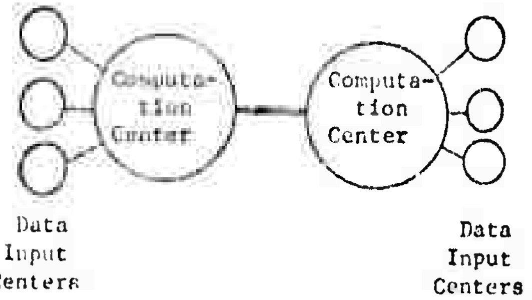


Figure A-3. Regionalized Computation System

Treatment of Acknowledgement Messages

One technique for the control of error and reliability in common practice is the use of acknowledgement messages. In systems employing only positive acknowledgement messages, accurate receipt of a message at a node automatically generates a return message to the transmitting node indicating that the message was correctly received and retransmission is unnecessary. Messages are periodically transmitted by the sending nodes until an acknowledgement is received. In other networks, an error in message transmission may generate a negative acknowledgement which causes retransmission of the message.

We will assume a perfectly functioning positive acknowledgement system, i.e., each message generates an acknowledgement along the same duplex line in the opposite direction, and no retransmissions are necessary.

With subscripted message sizes, equation (1) contains separate terms for the average queueing delay, $\lambda_i/[\mu_i C_i(\mu_i C_i - \lambda_i)]$, and transmission time, $1/\mu_i C_i$, of a message on link i. Of course, message transmission time is independent of consideration of acknowledgement messages, being a function only of the link's channel capacity and the size of the transmitted message. Thus transmission delay should now be $1/\mu_i' C_i$.

Delay time on queue, however, is a function of the total load on the system, including acknowledgement messages. Thus, the overall mean queueing delay on line i is still $\lambda_i/[\mu_i C_i(\mu_i C_i - \lambda_i)]$ where the unprimed variables reflect the arrival rates and sizes of all messages, acknowledgements included.

The weighting factor for communication delays, λ_i/γ, is chosen to reflect delays for the messages of interest; the particular choice depends on the objectives of the analyst when R is used as a criterion for optimization. Note that by using λ_i rather than λ_i', we will be weighting delays according to the flow of content messages only on each link, rather than all messages combined, a distinction not made by Kleinrock. Since, for our performance model, we are interested only in delays encountered by content messages, we will make this change. (γ, of course, must also reflect only content messages.) Thus, for a link which carries only acknowledgement messages from node i to node j, the contribution to the overall response time is zero, a situation which reflects our interest in the delays encountered by content messages only.

With the changes for variable message sizes and a different treatment of acknowledgement messages, the equation for communication delay as used in the CACTOS studies is given by

$$T_{Comm} = \sum_{i=1}^{M} \frac{\lambda_i'}{\gamma}\left(\frac{1}{\mu_i' C_i} + \frac{\lambda_i/\mu_i C_i}{\mu_i C_i - \lambda_i} + \frac{L_i}{\nu} + K\right) + K \qquad (2)$$

The Computer Throughput Model

Response time in a computer network depends on the processing rate of its computers as well as the processing rate of its communication facilities. The computation of the effective processing rate of a computer is an extremely complex problem, being a function of at least hundreds of hardware and software parameters. Many approaches to the problem of estimating computer throughput have been attempted, some of them involving step-by-step discrete-state simulation and some involving the construction of analytical models. The requirements of the CACTOS program dictated that the computer throughput model be fast enough that the answers are received virtually instantaneously, simple enough that the user inputs are minimal, and yet detailed enough that computation parameters might be meaningfully "traded off" with communication parameters. Speed and simplicity requirements quickly eliminated discrete-state simulation as a potential technique.

The question whether or not a simple and meaningful analytical model of computer throughput can be constructed is a moot one and depends mainly on the model's intended applications. For trade-off studies of the scope and generality of the CACTOS program, the analytical approach taken here was adequate. Moreover, it is felt that the approach, whereby such analytical models may be fairly readily constructed, is at least as important as the results. A small number of relevant hardware and software parameters was selected for the CACTOS model, but the approach is of sufficient generality

and open-endedness that different parameters, and more of them, might be similarly included as the analyst requires.

This approach presupposes a reference hardware configuration with known throughput parameter values against which other configurations are to be compared. Any standard configuration could be used as a reference; let us arbitrarily select an IBM 360/50 with model 2314 disc units and 512K bytes of core. The list of hardware and software parameters which we wish to include in the model is shown in Table 1; as just pointed out, this list is arbitrary and could be easily amended to suit a user's particular needs.

Although there is much discussion as to what the proper units of throughput should be, we will adopt the fairly artificial unit of modified bits/millisecond, comparable to the modified bits/second used by Roberts in his studies of trends in the costs of computer throughput.[6] Thus, a job's size in this model is described in units of modified bits, which, when divided by the effective processing rate output by the computer throughput model, yields the amount of time the job would consume on the hardware configuration in question.

Before constructing a throughput model, we will need some definitions:

- TP = throughput (effective processing rate for the hardware-software combination under consideration).
- T_{CPU} = computation time.
- T_{IO} = input-output time.
- f = fraction of a job's total time which is spent in computation (as opposed to I/O) if it is run on the reference hardware.
- n = fraction of CPU time overlapped by I/O.
- v = time a job's I/O takes when it is overlapped divided by the time the same I/O takes when performed sequentially.

Figure 4 shows a typical CPU-I/O cycle in various degrees of overlap which should clarify the preceding definitions. Two things should be noted here. One is that the range of v is from 1/c to 1 where c is the number of I/O channels, since with full utilization of all channels the I/O time could not be less than T_{IO}/c. Also note that n, the degree of I/O-CPU overlap, is associated with the job alone and is independent of the hardware under consideration. The rationale is that a given job is organized in a particular way, such that it issues (or can issue) an I/O command at a given point in its computation sequence, regardless of the hardware configuration upon which it is run.

TABLE A-1. COMPUTER THROUGHPUT MODEL INPUT PARAMETERS

Hardware	Software
Instruction rate	Ratio of computation time to total time consumed
Word size	
Primary memory size	CPU - I/O overlap
Peripheral descriptors	I/O - I/O overlap
• Average access time	
• Transfer rate	
• Maximum amount of information which may be transferred on one access (e.g., cylinder size for a disc)	

Figure 4 also gives us a clue as to how to go about estimating throughput. Since the processing rate is inversely proportional to the required time for a given unit of work (we use the modified bit), we need only add up the times shown in Figure 4c and invert to get a processing rate. Thus

$$\frac{1}{TP} = (1-n)\ T_{CPU} + n\ T_{CPU} + vT_{IO} - n\ T_{CPU}$$

which simplifies to

$$\frac{1}{TP} = (1-n)\ T_{CPU} + v\ T_{IO} \qquad (3a)$$

Equation (3a) is valid when I/O operations are not completely overlapped by computation, a condition expressed algebraicly by $nT_{CPU} < vT_{IO}$.

When $nT_{CPU} \geq vT_{IO}$, I/O is completely overlapped by computation, a condition illustrated in Figure 5. In this case, the CPU rate is the sole factor determining throughput and we must use the equation

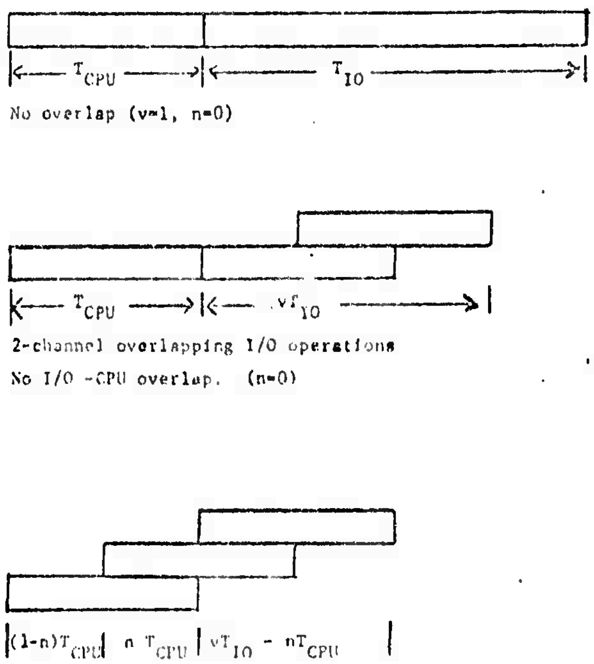
$$\frac{1}{TP} = T_{CPU} \quad , \quad nT_{CPU} \geq vT_{IO} \qquad (3b)$$

Equations (3a) and (3b) constitute a throughput model once we have a way to compute T_{CPU} and T_{IO} for the machine and workload under consideration.

First, if we assume that a computer's CPU is capable of processing p instructions/ms and that an instruction is capable of modifying w bits (w is generally the computer's word size), then the time per modified bit is $\frac{1}{wp}$. For a given job with a fraction f of its total time

spent in computation, we have

$$T_{CPU} = \frac{f}{wp} \ .$$



(a) No overlap (v=1, n=0)

(b) 2-channel overlapping I/O operations
No I/O -CPU overlap. (n=0)

(c) 2-channel overlapping I/O operations
I/O - CPU overlap

Figure A-4. A Typical Compute - I/O Cycle in Varying Degrees of Overlap (CPU time/total time = f = 1/3)

The calculation of T_{IO} is somewhat more difficult. First, we need to convert the units of I/O work into the equivalent amount of work in modified bits. Also, it is non-trivial to assess the effects of the size of primary memory on I/O time.

We begin by asserting that a job's I/O time is proportional to the number of I/O accesses, the average duration of each access, and the proportion of the job's time spent in I/O operations, i.e.,

$$T_{IO} = k \text{ (number of accesses) (average access duration) } (1-f)$$

The number of accesses required is a function of the primary memory size: the greater the memory size of the computer under consideration, the smaller the number of required accesses. We will assume that the number of accesses is inversely proportional to some power α of the memory size. Then, since the reference hardware has $5.12 * 10^5$ bytes of core memory, one access on the reference hardware would correspond to $\left(\frac{5.12*10^5}{m}\right)^{\alpha}$ accesses on a machine with m bytes of core memory. From a large amount of timing data for a real program on many hardware configurations, α has been estimated to have the value 0.83. The determination of α will be fully documented as part of a forthcoming document on validation of the CACTOS model.

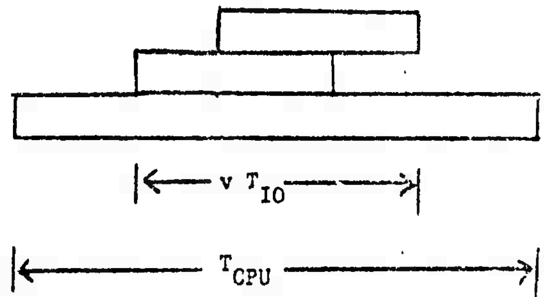


Figure A-5. A Compute-I/O Cycle in which I/O is Completely Overlapped by Computation

The average duration of an I/O operation is given by $a + \frac{r}{x}$, where a = average access time, r = average record size, and x = transfer rate of the storage device. But we know that a larger primary memory would permit the construction of larger records for secondary memory, a strategy which permits a gain in I/O efficiency. If we assume that primary memory size and secondary memory record size are, in fact, proportional, then the duration of an I/O operation on a machine with m bytes of core memory would be $a + \left(\frac{5.12*10^5}{m}\right)\left(\frac{r}{x}\right)$. A final point here is that little is gained if we are operating on a device which can handle only a limited amount of information without making another access. On a disc, it is not particularly beneficial to increase the record size beyond a cylinder's capacity. The implementation of the CACTOS model is cognizant of this and does not adjust the record size beyond that of the cylinder capacity or comparable quantity on the I/O device under consideration.

If we now let C = cylinder size, we can write the expression for I/O time:

$$T_{IO} = k\left(\frac{5.12*10^5}{m}\right)^{\alpha}\left(a + \frac{mr}{(5.12*10^5)\,x}\right)(1-f),$$

$$m \le \frac{5.12*10^5\,C}{r}$$

$$T_{IO} = k\left(\frac{r}{C}\right)^{\alpha}\left(a + \frac{C}{x}\right)\left(1-f\right),\quad m > \frac{5.12*10^5 C}{r}$$

We need only evaluate k to complete our computer throughput model. To do this, consider a job which is half computation and half I/O running on the reference hardware, with record size equal to the 2314 track size. We know that for this job and hardware configuration, $T_{IO} = T_{CPU}$, or

$$\frac{f}{wp} = k\left(5.12*10^5\right)^{\alpha}\left(a + \frac{mr}{(5.12*10^5)\,x}\right)(1-f)$$

Upon substituting $f = 0.5$, $m = 5.12*10^5$, and the manufacturer's published figures for w, p (we use the inverse of the add time but it may be more desirable to use the instruction rate for a typical instruction mix), a, r, and x, we may easily solve the equation to get

$k = 1.88*10^6$.

Integration of the Parts

To complete the whole computer network model, we need to do three more things: compute message delay from the packet delay given by equation (2), compute computational delay using the output of the computer throughput model, and sum the average communication and computation delays.

Large messages are not generally transmitted through a network in one piece but are divided into smaller packets which may be more readily handled. The packets are sent separately through the network and reassembled at the destination node. It is not adequate to treat this procedure in the model by dividing the average message sizes by the number of packets/message and multiplying the arrival rate by the same number. Instead, one must consider the actual distribution of message sizes.

If the cumulative distribution function describing message sizes is $Q(X) = Pr\{1/\mu_i < X\}$, then the fraction of messages of size less than X is simply Q(X). If Z is the maximum number of bits in a packet, then, by allowing X to assume the values Z, 2Z, 3Z, ..., corresponding to 1, 2, 3, ..., packets, we may easily compute the number of messages requiring X packets for transmission, which is

$$[Q(X) - Q(X-1)]\lambda_i' ,\; X = 1,2,3,\ldots$$

and

$$Q(0) = 0.$$

The total number of packets over each link i in the network may then be readily calculated and replaces λ_i' in equation (2). (λ_i must also be adjusted to reflect packet traffic.) Dividing the total number of bits transmitted by the number of packets required gives the new average content message size, $1/\mu_i'$; again, μ_i must be appropriately adjusted in the straightforward way.

Table 2 shows how the number of packets and average packet size are calculated for a sample of 1000 messages with an average message size of 100 bits and a maximum packet size of 100 bits. In this case, an expected 1578 packets, of average size 63.4 bits, would be required.

TABLE A-2. CALCULATION OF THE NUMBER AND AVERAGE SIZE OF PACKETS WITH AVERAGE MESSAGE SIZE = 100 BITS AND MAXIMUM PACKET SIZE = 100 BITS

No. of Packets Per Message	Message Size Range (Bits)	Q(X)	No. of Messages	No. of Packets
1	1-100	.6321	632	632
2	101-200	.8647	233	466
3	201-300	.9502	86	258
4	301-400	.9812	31	124
5	401-500	.9933	12	60
6	501-600	.9975	4	24
7	601-700	.9991	2	14
Total			1000	1578
Average Packet Size = $\frac{1000*100}{1578}$ = 63.4 bits				

Two questions about statistical validity arise as the result of abandoning the message as the individual atom being transmitted through the network and treating messages as groups of smaller amounts of information, called packets.

Important to the calculations is the assumption that message arrivals are independent of message lengths, an assumption discussed at length by Kleinrock.[3] When considering long, undivided messages arriving at nodes, it is clear that this assumption becomes less valid, since the minimum interarrival time between long messages must be affected by the long

transmission times associated with them.* Thus, one might expect an improvement in statistical validity by treating long messages as groups of shorter ones.

On the other hand, the introduction of the concept of packets in the manner described creates other perturbations which may affect the calculations even more. The derivation of equation (2) assumes that the sizes of the units of information being transmitted through the network are taken from an exponential distribution. This is not likely to be the case when packets are used, because packet sizes will never exceed the maximum allowable packet size. As Kleinrock points out,[5] however, there is an easy treatment of this dilemma by resorting to the Pollaczek-Khinchin formula for channel delay with any message-length distribution of known mean $1/\mu_i$ and variance σ_i^2:

$$\text{delay on channel } i = \frac{2 - \frac{\lambda_i}{\mu_i C_i}\left(1 - \mu_i^2 \sigma_i^2\right)}{2\left(\mu_i C_i - \lambda_i\right)} \qquad (4)$$

Although we have not done it for either system under study, equation (4) could easily be incorporated into the response time equations, and μ_i and σ_i could then be determined from analysis of the system's packet traffic. While this would end the assumption of all random processes within the system being governed by negative exponential distributions, it might be a better approximation to the true performance of a packet system than is represented by equation (2).

The second question of validity concerns the distribution of the arrival time of messages and jobs at the computation nodes. If one considers the arrival of messages at destination nodes, where a message consists of some number of packets which make their way through the system, then both theoretical considerations[7] and measurements on analogous systems[2] suggest that a gamma distribution best describes message arrivals at terminal nodes.

To explore this possibility a bit further, assume now that packet arrival in the system is a Poisson process with mean arrival rate λ_p. (We know that at a given source node, this would be a terrible assumption because packets arrive in groups which constitute a message, followed by a pause until the next message has been constituted and received. But again, if we consider a large system, we can make an assumption similar to Kleinrock's and say that interarrival times for the system as a whole are independent of both transmission times and blocking considerations and thus constitute a Poisson process.)

Packet interarrival times, then, are governed by an exponential distribution whose cumulative distribution function is

$$Q(t) = 1 - e^{-\lambda_p t}$$

If there is an average of n packets per message, then the gamma distribution describing message arrivals is

$$Q(t) = \frac{\lambda_p^{\,n}}{\Gamma(n)} \int_0^t t^{n-1} e^{-\lambda_p x}\, dx \qquad (5)$$

where

$$\Gamma(n) = \int_0^\infty x^{n-1} e^{-x}\, dx$$

Further evidence of the credibility of the gamma distribution here is obtained from consideration of the special case where messages are not divided into packets; i.e., n = 1. This leads to

$$Q(t) = \frac{\lambda_p}{\Gamma(1)} \int_0^t e^{-\lambda_p x} dx = 1 - e^{-\lambda_p t}$$

which is the exponential distribution used for the arrival of one-packet messages by Kleinrock in the derivation of the original model.

The queueing analysis of computation jobs at network nodes depends on the assumption that message arrivals at these nodes constitute a Poisson process. If a gamma distribution, rather than an exponential distribution, describes these arrivals, the model is obviously inaccurate when messages are split into packets. Unfortunately, a mathematical analysis of queueing and response times based on gamma statistics does not appear tractable, so the ramifications

* But, as Kleinrock points out, this effect is minimized when a large system (many source nodes) is considered because arrivals at one node are independent of message lengths at other nodes, and thus the overall arrival rate into a large system tends to approximate independence of all message lengths, a conclusion well substantiated by simulation results.

f this development can probably be deduced only rom a simulation. Fuchs and Jackson, who ound by measurement that arrival statistics n four time-sharing systems were gamma istributed, addressed themselves to errors nduced by substituting exponential distribu- ions.[2] Differences between the cumulative istributions were plotted as a function of the oefficient of variation of the gamma distribu- ion, but these say little about how they elate to errors in the final outputs of the odel. We will continue to treat packet rrivals as a Poisson process and assume that hat errors thus induced are small.

After making the necessary changes to ompute message delay by computing the number of packets required, which we will call b, and ultiplying the message arrival rate by b, T_{COMM} (communication delay) may be properly computed rom equation (2).

The computational delay at a node i may e computed from the simple single-server Markovian arrival queueing formula

$$T_{COMP} = \frac{1}{(\sigma_i * TP) - \lambda_i^J}$$

where σ_i = mean job size at node i

TP = throughput rate as calculated from equation (3)

λ_i^J = mean job arrival rate at node i

The overall average computational delay may be computed from the weighted average

$$T_{COMP} = \sum_{i=1}^{N} \frac{\lambda_i^J}{\gamma} \left(\frac{1}{(\sigma_i * TP) - \lambda_i^J} \right)$$

where N = number of nodes in the network and

$$\gamma = \sum_{i=1}^{N} \lambda_i^J \quad \text{= total job input rate for the network}$$

Finally, if we define $\times$ to be the number of remote jobs divided by the total number of jobs (remote jobs appear in positions other than the diagonal of the job arrival matrix, and are the only ones which require inter- node communication) and remember that two messages are associated with each remote job, the overall average response time for the system may be computed from

$$T = 2RT_{COMM} + T_{COMP}.$$

SUMMARY

In this paper, the need for quantitative modeling of computer networks has been discussed, and one approach to the construction of an analytical model of computer network performance has been outlined. The validation of the model and some results obtained by using it in cost- effectiveness trade-off studies are to be topics of future publications.

The definitions of parameters in this paper have been given more from the point of view of the research scientist than from that of the system designer. The research scientist is interested in having the parametric description of a given job remain invariant over all hard- ware configurations; therefore, such job para- meters as the ratio of CPU time to total time and the degree of I/O - I/O overlap always refer to the performance of the job on the reference hardware. To the system designer, it may be inconvenient to have to describe a real job in terms of its behavior on a config- uration on which it has never been run. It is possible, however, to develop formulas for the translation of parameters measured on a known system to their corresponding values on the reference configuration, and it is not diffi- cult to follow the steps outlined in this paper and redevelop the computer throughput model using some other reference configuration which is more convenient to the user. Thus, in a broad sense, the results presented here should be of interest both to the generalist and the specialist. It is anticipated that the model will be a useful tool in the evaluation of proposed changes to existing networks, as an aid in the design of new networks, and in understanding the behavior of computer networks in more general ways.

BIBLIOGRAPHY

1. Frank, H., Minimum Link Traffic Routing, unpublished, 1970.

2. Fuchs, E., and P. E. Jackson, "Estimates of Distributions of Random Variables for Certain Computer Communications Traffic Models," Communications of the ACM, December, 1970.

3. Kleinrock, L., Communication Nets: Stochastic Message Flow and Delay, McGraw-Hill, New York, 1964.

4. Dijkstra, E. W., "A Note on Two Problems in Connection with Graphs," Num. Math., Vol. 1, 1959.

5. Kleinrock, L., "Analytic and Simulation Methods in Computer Network Design," Proceedings of the SJCC, Atlantic City, N.J., 1970.

6. Roberts, L. G., Data Processing Technology Forecast, Unpublished, 1969.

7. Hahn, G. J., and S. S. Shapiro, Statistical Models in Engineering, Wiley and Sons, New York, 1967.

APPENDIX B. VALIDATION OF THE CACTOS MODEL

Three major parts of the CACTOS model were addressed in the validation of the model equations. These were:

- Communication analysis equations
- Hardware computation analysis equations
- Software computation analysis equations

The communication analysis methodology was based on the work of Kleinrock [1] and others; Kleinrock discusses validation of the communication analysis. In addition, the Project conducted further validation experiments using a discrete simulation model based on ECSS, a computer simulation language in SIMSCRIPT developed at the RAND Corporation.

The hardware computation validation consisted of examining the equation for computer throughput based on various hardware parameters. This throughput equation was parameterized in that the exponent of core memory was undefined. The reason for this was that the contribution of the other hardware features was better defined. To perform the validation, a set of processing runs from a variety of configurations for the same programs was needed. One program that exactly satisfies this criterion is the IBM Sort/Merge program. Twenty configurations were used; they are shown in Table B-1. Calibration of the parameter associated with core storage was performed on the fifth configuration.

The range of dispersion in percentage varied from 2%-45%. In three cases, the dispersion exceeded 23%. For these cases, the situation was small core size with 2311 and 3330 disk units. The fit improved as core increased. This was most important, since the experiments involved larger core sizes than those in Table B-1.

TABLE B-1. COMPUTER CONFIGURATION FOR VALIDATION RUNS

Configuration	Computer	Core	Disk
1	360/30	38K	2311
2	360/50	44K	2311
3	360/50	44K	2314
4	360/50	100K	2311
5	360/50	100K	2314
6	360/50	200K	2311
7	360/50	200K	2314
8	360/65	100K	2311
9	360/65	100K	2314
10	360/65	200K	2311
11	360/65	200K	2314
12	370/155	44K	2311
13	370/155	44K	2314
14	370/155	44K	3330
15	370/155	100K	2311
16	370/155	100K	2314
17	370/155	100K	3330
18	370/155	200K	2311
19	370/155	200K	2314
20	370/155	200K	3330

It can be noted that this represents a shortcoming of an analytic approach versus a discrete one, in that accuracy across wider ranges of parameters is possible in the discrete case. Had the experiments been oriented toward a close fit at every core size, several equations could have been employed.

The third part of the validation of the software aspects of computation includes record size and the relationship between CPU and I/O in terms of overlap and balance. To validate this, the JOVIAL program shown in Figure B-1 was constructed. The purpose of the program is to carry out a specified number of CPU and I/O operations while timing itself internally. (JOVIAL permits such timing.) For a variety of I/O and CPU balances and overlaps, the results of the model and program were compared. The results are summarized in Figure B-2. In this figure the horizontal axis is the experiment number while the vertical axis is the time in seconds. The points labeled with an X are those of the program. The program was run on a 370/155. The dispersion for all but one case is less than 20%. Since the direction of the times and incremental rise for both the program and the model was the same, this was felt to be adequate.

```
     START
     DEFINE TTY ''2'' $
    ITEM TTIME F $
    ITEM NUMBER F $
    ITEM IJK I 32 S $
     ITEM FLOAT F $
     ITEM AI F $
     ITEM BI F $
     ITEM CI F $
     ITEM RI F $
     ITEM TI F $
     ITEM LLOOP I 32 S $
     ITEM NLOOP I 32 S $
     ITEM KK I 32 S $
     ITEM KI I 32 S $
     ITEM KJ I 32 S $
     ITEM IOCOUNT I 32 S $
     ITEM TIME I $
     ITEM II F $
     ITEM JI F $
     ITEM QI F $
     ITEM CS F $
     ITEM IO F $
     ITEM TEMP I 32 U $
     ITEM TEMPL I 32 U $
     ITEM CONSTANT F $ '' NO. LOOPS/SEC.''
  TABLE SPRST R 2 S 9 $
   BEGIN
     ITEM  SFCDE I 32 U 0  0 N $
     ITEM  SRST2 I  8 U 1 16 M $
     ITEM  SWAIT I  1 U 1  8 D $
     ITEM  SSTS  I  8 U 1 24 M $
     ITEM SCLS1 I 1 U 1 11 D $
     ITEM SCLS2 I 1 U 1 15 D $
     ITEM  SDATA I 32 U 3  0 N $
     ITEM  SREC  I 16 S 4  0 M $
     ITEM  SBYTS I 16 U 4 16 M $
     ITEM  SPLOC I 24 U 5  8 M $
     ITEM  SLENG I 16 S 6 16 M $
     END
  TABLE BUFFER R 2000 S 1 $
   BEGIN
     ITEM  BUF1  I 32 U 0  0 N $
   END
  TABLE BUFFET R 2000 S 1 $
   BEGIN
     ITEM  BUF2  I 32 U 0  0 N $
   END
     TABLE CRT  R 2 S 20 $                '' FILES ''
      BEGIN
     ITEM  CNWDS I 8 U 0 0 M $
     ITEM  RUSTS I 8 U 0 8 M $
     ITEM  KEYLG I 8 U 0 16 M $
     ITEM  STATS I 8 U 0 24 M $
     ITEM  FWTPT I 32 U 1 0 N $
     ITEM  ORGAN H 2 2 0 M $
     ITEM  SECUR H 1 2 16 M $
     ITEM  FORMS I 8 U 2 24 M $
     ITEM  SIZES I 32 U 3 0 N $
     ITEM  BLOCK I 4  U 4 0 N $
```

Figure B-1. JOVIAL Model Validation Program

```
          ITEM  SPMBF I 24 U 4 8 M $
          ITEM  ACCES H 1 5 0 M $
          ITEM  PRTCT H 1 5 8 M $
          ITEM  VOLTP I 8 U 5 24 M $
          ITEM  PRVTE I 8 U 6 0  M $
          ITEM  DENSE I 8 U 6 8  M $
          ITEM  DPRIV I 8 U 6 16 M $
          ITEM  VOLNO H 4 7 0 N $
          ITEM  WORDB I 32 U 8 0 N $
          ITEM  NAMES H 8 9 0 N $
          ITEM  NAMEI I 32 U 10 0 N $
          ITEM  FID1 I 32 U 11 0 N $
            BEGIN  0  0  END
          ITEM  FID2 I 32 U 12 0 N $
            BEGIN  0  0  END
          ITEM  FID3 I 32 U 13 0 N $
            BEGIN  0  0  END   END
  D1.     REDT(8H(A)        ),-1,1,0,'LOC(AI))$
          REDT(8H(B)        ),-1,1,0,'LOC(BI))$
          REDT(8H(C)        ),-1,1,0,'LOC(CI))$
          REDT(8H(T)        ),-1,1,0,'LOC(TI))$
         REDT(8H(NUMBER   ),-1,1,0,'LOC(NUMBER))$
          RI=3200.0$
          CONSTANT=310000.0$
          FOR I=0,1,1 $
          BEGIN
          '' OPEN 2 TEMPORARY DISC FILES ''
          CHNDS($I$)=20 $
          MDSTS($I$)=1  $
          ORGAN($I$)=2H(SI) $
          FORMS($I$)=1 $
         IJK=NUMBER $
          BLOCK($I$)=1 $
          SPMBF($I$)='LOC(BUFFER)$
          SIZES($I$)=RI    $
          PRTCT($I$)=1H(N)$
          VOLTP($I$)=0 $
          PRVTE($I$)=2 $
          DPRIV($I$)=1 $
          VOLNO($I$)=4H(0000) $
          NAMES($0$)=8H(TMPORARY) $
          NAMES($1$)=8H(TMPESTUS) $
          IF NAMES($I$) EQ 8H(TMPORARY) $
              BEGIN
               ADPCAT(1,'LOC(CRT)) $
               CATCHK('LOC(CRT))    $
              END
          IF NAMES($I$) EQ 8H(TMPESTUS) $
              BEGIN
               ADPCAT(1,'LOC(CRT)+80) $
               CATCHK('LOC(CRT)+80)    $
              END
              END
         FOR L=1,1,IJK $
         BEGIN
          IO=(87.5*(RI/312.0))/2000.0 $
          IF AI EQ 1.0$ BEGIN  CS=TI $  GOTO D3$ END
          CS=((AI*IO)/(1.0-AI))$
  D3.     II= I( CS-BI*IO)*CONSTANT) $
          IF II LS 0.0$  II=0.0$
```

Figure B-1. JOVIAL Model Validation Program (Cont'd)

```
          JI= (T0*(1.0-CI)*CONSTANT) $
          QI= (TI*(1.0-AI))/(10*2)   $
          IOCCUNT=0 $
          DIRECT
          LA     0,1
          SVC    19
          LBR    10,LLOOP
          ST     3,LLOOP(0,10)
          JOVIAL
    D4.   NLOOP=0 $
          FLOAT=RANDOM( ) $
          KI=II*LOGNAT(FLOAT)$
          FLOAT=RANDOM( ) $
          KJ=JI*LOGNAT(FLOAT)$
          IF KI LS 0$  KI=KI*(-1)$
          IF KJ LS 0$  KJ=KJ*(-1)$
          KK=KI $
    D5.   NLOOP=NLOOP+1 $
          IF NLOOP LS KK$ GOTO D5$
          IF KK NQ KI$ GOTO D6$
          SETUP(0)$
          ADPSPAM(5,TEMP)$  ''POSITION ''
          SWAIT($0$)=0 $ ''NO WAIT ON WRITE''
          ADPSPAM(2,TEMP) $  ''WRITE''
          SETUP(C)$
          SWAIT($0$)=0 $  '' NO WAIT ON CHECK ''
          ADPSPAM(7,TEMP) $ ''CHECK(REQUIRED BY SPAM ''
          KK=KI+KJ $
          GOTO D5$
    D6.   TEMP1=TEMP+36$''SET FOR 2ND ENTRY OF SPRST''
          SETUP(1)$
          ADPSPAM(5,TEMP1) $   ''POSITION ''
          SETUP(1)$
          ADPSPAM(1,TEMP1) $   ''READ (WAITING) ''
          IOCCUNT=IOCOUNT+1 $
          IF IOCCUNT LS QI $ GOTO D4$
          DIRECT
          LA     0,1
          SVC    19
          LBR    10,LLOOP
          S      3,LLOOP(0,10)
          ST     3,LLOOP(0,10)
          JOVIAL
         TTIME=LLOOP/60.0 $
         PRNT(1,TTIME,4H(F    ),0,-1,0,0,-1,1) $
         TIME=TIME+TTIME $
         END
         TIME=TIME/NUMBER $
          PCSN=0 $
          PRNT(1,TIME,4H(F    ),0,-1,0,0,-1,1)$
          PRNT(1,QI,4H(E    ),0,-1,0,0,-1,1)$
          PRNT(1,T0,4H(E    ),0,-1,0,0,-1,1)$
          PRNT(1,CS,4H(E    ),0,-1,0,0,-1,1)$
          PRNT(1,II,4H(E    ),0,-1,0,0,-1,1)$
          PRNT(1,JI,4H(E    ),0,-1,0,0,-1,1)$
          CLOZ('LCC(CRT))$
          STOP  D1 $
       PROC RANDOM $
          ITEM  RANDOM  F  $
          BEGIN
```

Figure B-1. JOVIAL Model Validation Program (Cont'd)

```
         DIRECT
         BALR   10,0
         USING  *,10
         L      1,SEED
         SR     0,0
         CR     0,1
         BC     6,LAB
         LA     0,1
         SVC    19
         LR     1,0
LAB      M      0,MULT
         ST     1,SEED
         LD     0,FLOT
         AD     0,ZERO
         BC     15,OUT
ZERO     DC     0
FLOT     DC     X(46000000)
SEED     DC     0
MULT     DC     32781
OUT      STE    0,RANDOM(0,13)
         JOVIAL
         END
         PROC SETUP(GROUP)$
         ITEM  GROUP I 32 U $   BEGIN
         SREC($GROUP$)=1$
         SDATA($GROUP$)='LOC(SPLOC) $
         SPLOC($GROUP$)='LOC(BUFFER)$
         SLENG($GROUP$)=RT$
         SWAIT($GROUP$)=1$ ''WAIT''
         SFCDE($GROUP$)=FWTPT($GROUP$)$
         SCLS1($GROUP$)=1 $
         SCLS2($GROUP$)=1 $
         END ''SETUP''
         TERM $
```

Figure B-1. JOVIAL Model Validation Program (Cont'd)

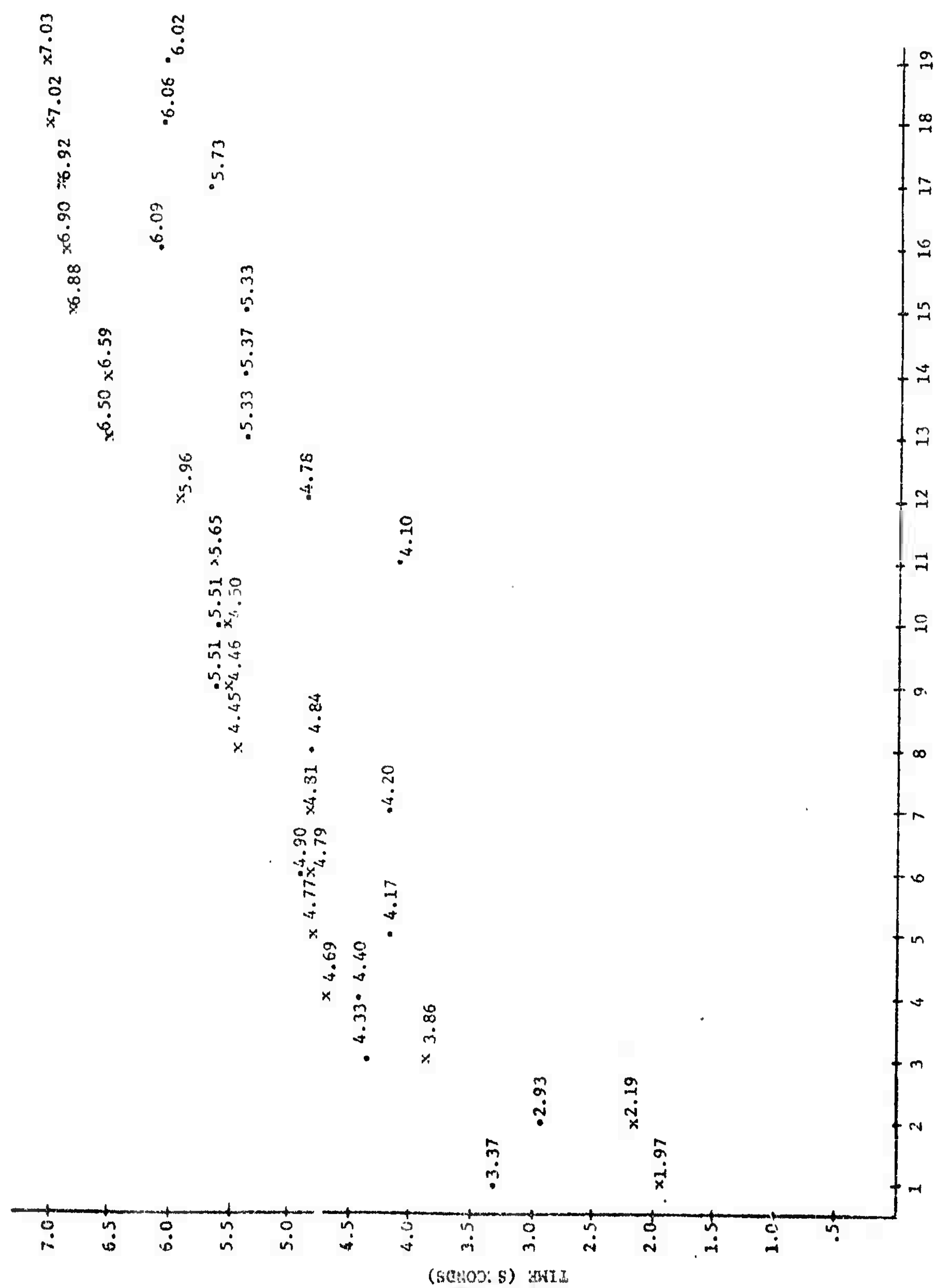


Figure B-2. Comparison of CACTOS Model and Validation Program for CPU Overlap and Balance